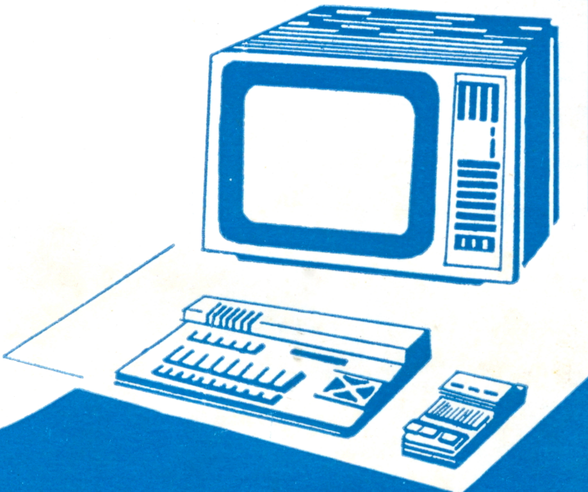
CEC—I

韩仲清　　廖兴祥　编著



·如何使用
中华学习机

成都三开元电脑部

如何使用中华学习机

韩仲清　廖兴祥　编著

成 都 三 开 元 电 脑 部

内　容　提　要

本书介绍了中华学习机CEC-I的主要部件及其安装；BASIC程序设计的基本概念和方法；顺序文件、随机文件的建立和检索；操作系统和汉字系统的用法；如何绘图和编制唱歌程序；LOGO语言及海龟作图法。

该书按教材形式编写，文字通俗，示例浅显，内容丰富实用。适合青少年和家长们自学，可作中小学、少年宫、中专、职业班及干部专修班等学生的教材，也可供从事APPLEⅡ、紫金Ⅱ教学和科研的人员参考。

本书由韩仲清统稿，张陞楷审校。

前　　言

在不久的将来，电子计算机这位神秘的伴侣将同电视机一样很快走进寻常人家，成为人们工作、学习和生活中不可缺少的工具。特别是价廉物美的适宜于家庭和青少年使用的中华学习机的普及和推广，将大大地加快这种进程。为四化进程添加一点催化剂，这是我们编写《如何使用中华学习机》的主要目的。

本书着眼于广大的青少年、中小学生和计算机的初学者。因而本书没有涉及高深的数理知识，大部份示例基于中学数学的内容和水平。为便于自学和家长辅导自己的孩子学习计算机知识，书中附有大量的例子，对初学者学习和使用中容易发生问题的地方，尽量给予详细注解和说明。大部份章节末尾给出了问题与思考，供读者回味所学内容，每章都提供了习题，供读者课余练习，以巩固和加深所学的知识，提高解决问题的能力。

在文字上力求通俗易懂，准确、生动。尽量符合中小学生的口味和习惯。但我们毕竟不是中小学生，只是过来人。只能依稀凭借童年的美好回忆，尽量与孩子们为伍。

在编写本书的过程中，得到成都三开元电脑技术开发部舒新生等同志的大力支持和帮助，四川大学计算机科学系张陞楷副教授审阅了全部书稿，在此一并表示感谢！

虽然作者尽了努力，但书中定有不妥之处，请读者和专家指正。

韩仲清

1988年6月5日

于四川大学计算机科学系

目　　录

第一章　中华学习机的主要部件及其安装……………… 1

1.1　主要部件及其作用……………………………… 2

1.1.1　主机 ………………………………………… 2

1.1.2　键盘 ………………………………………… 3

1.1.3　显示器……………………………………… 5

1.1.4　打印机……………………………………… 7

1.1.5　软盘和软盘驱动器………………………… 7

1.1.6　录音机……………………………………… 9

1.1.7　电源 ………………………………………… 9

1.2　部件之间的线路连接…………………………… 9

1.2.1　主机和屏幕的连接………………………10

1.2.2　主机和录音机的连接……………………13

1.2.3　主机和扩充卡的连接……………………13

1.2.4　主机和打印机的连接……………………15

1.2.5　主机和磁盘驱动器的连接………………15

1.2.6　主机和游戏操纵杆的连接………………16

1.3　系统的启动……………………………………16

1.3.1　基本系统的启动与自动进入BASIC语言 ………………………………………16

1.3.2　连接驱动器后系统的启动………………17

1.3.3　中西文方式的转换………………………18

1.3.4　系统的重新启动…………………………18

1.3.5　LOGO语言的进入方法…………………18

1.4 中华学习机的自动检测……………………………19
1.5 中华学习机的主要技术指标……………………21
习题 …………………………………………………23
第二章 BASIC语言的基本概念……………………24
2.1 什么叫BASIC语言………………………………24
2.1.1 BASIC是什么意思……………………………24
2.1.2 BASIC语言的特点……………………………25
2.1.3 BASIC语言的使用方式………………………26
2.1.4 本书用到的书写记号约定……………………27
2.2 BASIC语言用到的符号…………………………28
2.2.1 字符集…………………………………………28
2.2.2 BASIC语言的保留字…………………………29
2.3 常量、变量和字符串……………………………30
2.3.1 常量及其写法…………………………………30
2.3.2 变量 …………………………………………31
2.3.3 数据类型………………………………………32
2.3.4 数值的输出格式………………………………33
2.4 函数 ………………………………………………35
2.5 表达式………………………………………………36
2.5.1 运算符及其优先级别…………………………36
2.5.2 算术表达式……………………………………38
2.5.3 关系表达式……………………………………40
2.5.4 逻辑表达式……………………………………40
2.5.5 字符串表达式…………………………………42
2.6 BASIC程序…………………………………………42
2.7 BASIC程序的输入…………………………………45
2.7.1 如何进入BASIC系统 ……………………………45

2.7.2 如何输入程序……………………………………48
问题与思考……………………………………………52
习题……………（……………………………………52
第三章 简单的BASIC程序…………………………54
3.1 赋值语句………………………………………54
3.1.1 赋值语句的格式和作用………………54
3.1.2 赋值语句的执行步骤…………………58
3.1.3 使用赋值语句时的注意事项…………59
3.2 键盘输入语句…………………………………63
3.2.1 键盘输入语句的格式及其作用………63
3.2.2 使用INPU时注意事项………………66
3.2.3 GET语句…………………………………67
3.3 输出语句………………………………………68
3.3.1 输出语句的格式和作用………………68
3.3.2 分隔符“，”和“；”在PRINT中的
用法…………………………………………70
3.3.3 打印空行和换行打印…………………72
3.4 程序实例………………………………………74
3.5 常用命令和误操作的修改……………………76
3.5.1 常用命令…………………………………76
3.5.2 误操作的修改……………………………78
问题与思考……………………………………………79
习题……………………………………………………79
第四章 转向和分支程序…………………………83
4.1 转向语句………………………………………83
4.2 条件语句………………………………………85
4 2 1 条件语句的格式和作用………………85

4.2.2 条件语句的执行流程……………………97
4.3 读数据和置数据语句……………………………98
4.3.1 读数据和置数据语句的格式和作用…99
4.3.2 使用READ和DATA时注意…………103
4.4 数据指针的恢复……………………………104
4.5 框图法……………………………………………108
4.6 程序实例……………………………………………111
4.7 源程序如何打印和存盘…………………………117
问题与思考……………………………………………119
习题………………………………………………………119
第五章 循环程序……………………………………125
5.1 什么是循环……………………………………125
5.2 循环语句………………………………………125
5.2.1 循环语句的格式和作用………………125
5.2.2 循环语句的执行流程…………………131
5.2.3 使用循环语句时注意…………………131
5.3 多重循环………………………………………138
5.3.1 多重循环的结构…………………………138
5.3.2 使用多重循环时注意……………………140
5.4 循环程序实例……………………………………142
问题与思考……………………………………………148
习题………………………………………………………148
第六章 数组和维语句………………………………153
6.1 什么是数组……………………………………153
6.2 数组的表示方法………………………………154
6.2.1 维语句的格式……………………………154
6.2.2 使用维语句时注意………………………156

6.2.3 如何给数组赋值……………………………………158
6.3 数组的应用……………………………………………163
6.4 程序实例………………………………………………167
问题与思考………………………………………………172
习题………………………………………………………172
第七章 BASIC函数…………………………………………175
7.1 算术运算函数…………………………………………175
7.1.1 三角函数……………………………………………175
7.1.2 其它数学函数………………………………………176
7.2 字符串处理函数………………………………………179
7.3 数据类型转换函数……………………………………182
7.4 控制输出格式的函数和命令…………………………185
7.4.1 格式及其作用………………………………………185
7.4.2 打印格式的控制……………………………………188
7.5 自定义函数……………………………………………190
问题与思考………………………………………………193
习题………………………………………………………193
第八章 子程序……………………………………………195
8.1 转子语句和返回语句…………………………………195
8.1.1 语句格式及其作用…………………………………195
8.1.2 编写和调用子程序时注意…………………………202
8.2 子程序实例……………………………………………207
问题与思考………………………………………………211
习题………………………………………………………211
第九章 文件管理系统……………………………………217
9.1 文件的基本概念………………………………………217
9.1.1 数据、字段、记录、文件…………………………217

9.1.2 文件的种类……221
9.2 程序文件的建立和载入……223
9.3 顺序文件的建立和检索……225
9.3.1 顺序文件的建立方式……225
9.3.2 顺序文件的检索……230
9.3.3 顺序文件的修改……234
9.4 随机文件的建立和检索……236
9.4.1 随机文件的建立方式……236
9.4.2 随机文件的检索……240
9.4.3 随机文件的添加和修改……242
问题与思考……246
习题……246
第十章 BASIC程序的调试与设计技术……247
10.1 程序的调试……247
10.1.1 追踪程序的运行……247
10.1.2 取消追踪……248
10.2 出错处理语句……249
10.3 节省存贮空间……252
10.3.1 节省存贮空间的一些措施……252
10.3.2 BASIC的存贮分配情况……254
10.4 提高程序的执行速度……255
10.5 程序的链接运行……256
10.6 BASIC应用程序设计实例……258
习题……262
第十一章 BASIC绘图和音乐……265
11.1 BASIC低分辨率绘图语句……265
11 2 BASIC高分辨率绘图语句……268

11.3 高分辨率图形的向量作图法……………………272
11.4 BASIC音乐程序的编制……………………281
习题……………………………………………285
第十二章 操作系统概述……………………286
12.1 什么是操作系统……………………………286
12.2 操作系统的装入……………………………286
12.2.1 开机引导……………………………287
12.2.2 命令方式引导………………………287
12.3 操作系统的主要应用………………………287
12.3.1 新盘片的初始化……………………288
12.3.2 列文件目录…………………………289
12.3.3 保存BASIC程序……………………291
12.3.4 把BASIC程序调入内存……………292
12.3.5 复制磁盘文件………………………293
12.3.6 删除文件……………………………296
12.4 操作系统的主要命令及其作用……………297
12.4.1 管理磁盘文件的命令………………298
12.4.2 与BASIC程序有关的命令…………301
12.4.3 用于语言转换的命令………………302
习题……………………………………………303
第十三章 中华学习机的汉字系统…………304
13.1 汉字系统的功能……………………………304
13.2 汉字系统的进入与退出……………………304
13.2.1 汉字系统的进入……………………304
13.2.2 汉字系统的退出……………………305
13.3 汉字的输入方法……………………………305
3 3 1 输入法的选择………………………305

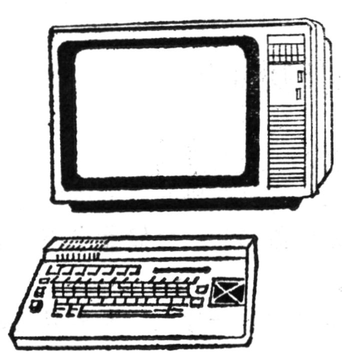
13.3.2 拼音输入法……306
13.3.3 区位码输入法……307
13.3.4 特殊符号的输入法……307
13.4 汉字的打印输出……308
13.4.1 设置打印汉字的字型号……308
13.4.2 设置字间距……309
13.4.3 设置行距……309
13.4.4 设置行允许字数……309
13.4.5 汉字打印实例程序……309
13.4.6 对某些打印机适应汉字打印的修改…310
习题……312
第十四章 LOGO 语言……313
14.1 LOGO语言概述……313
14.1.1 什么是LOGO语言……313
14.1.2 如何启动LOGO语言系统……315
14.1.3 LOGO的主要概念……317
14.2 LOGO的绘图功能……319
14.2.1 海龟作图的基本命令……319
14.2.2 过程……336
14.2.3 递归过程和条件语句……348
14.3 数、字和表处理功能……357
14.3.1 数和算术运算……357
14.3.2 字和字处理命令……358
14.3.3 表和表处理命令……359
14.4 数值计算功能……361
14.4.1 表达式……361
14.4.2 赋值命令和传送命令……364

14.4.3 如何编写数值计算程序……………………367
14.5 LOGO基本命令一览表…………………………369
14.6 海龟作图程序选编…………………………………372
14.6.1 利用正方形来构成的图形……………372
14.6.2 利用矩形来构成的图形………………374
14.6.3 利用正方形、矩形来构成的图形……376
14.6.4 利用三角形来构成的图形……………377
14.6.5 利用弧和圆来构成的图形……………379
14.6.6 变化距离和角度来构成不同的图形…381
14.6.7 用一个图形来构成其它图形…………383
问题与思考…………………………………………………388
习题…………………………………………………………388
附录Ⅰ BASIC出错信息表……………………………390
附录Ⅱ DOS 3.3出错信息表…………………………393
附录Ⅲ 区位码和汉字 符号对照表……………………396

第一章　中华学习机的主要部件及其安装

中华学习机是一种价格低廉，功能较强，结构灵活，使用简便，无特殊环境要求的微型电子计算机。该机迅速普及到中、小学，进入家庭，将有力地促进整个中华民族的进步和繁荣。

中华学习机的主机配上家用彩色或黑白电视机就可以组成最基本的微机系统。如果增配家用录音机（可以是廉价的盒式录音机）就构成了在录音磁带上保存程序和数据的中华学习机系统。再配上打印机及软盘驱动器就构成一台和Apple Ⅱe（苹果）机完全兼容，功能相当且在汉字处理能力方面有所增强的能用于中、小型计算与事务处理的计算机系统。如图1.1。这样的中华学习机系统其价格却约为Apple Ⅱe机之一半。



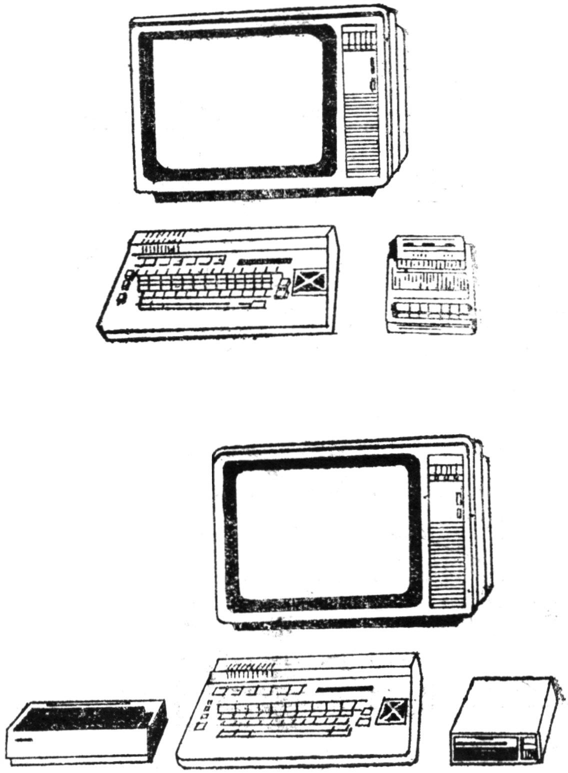


图1.1 三种配置的学习机系统

1.1 主要部件及其作用

1.1.1 主机

中华学习机的主机在机壳内（键盘下面）的主电路板上。它由中央处理器6502、存储器电路、输入输出接口电路和电源所组成。

中央处理器6502是一片大规模集成电路。它的时钟为1

MHz，一种为＋5V的电源。有8条数据总线，16条地址总线（寻址范围64K字节）。中央处理器6502是整个机器的核心。

存储器电路可分为半导体存储器和存储器管理部件。半导体存储器有RAM和ROM两种。由两片64K×4位的集成电路50464组成64K字节的RAM。由一片集成电路27256组成32K字节的ROM。ROM上固化有监控程序及BASIC语言。存储器管理部件由门阵列器件组成。在主电路板上，共使用了三片专用集成电路，除存储器管理部件外，还有一片门阵列器件为输入输出管理部件和一片可编程阵列逻辑电路PAL，其作用为产生时序信号及存储器行列地址选通信号等。

一般在主电路板上的扩充汉字系统插座里，插有一片固化有汉字系统的ROM片子27256、两片固化有汉字字库的ROM片子和两片TTL电路。

1.1.2 键盘

中华学习机的键盘安装在主机外壳上，具备有标准打字机键盘的各种键，键位的排列如标准打字机。见图1.2。

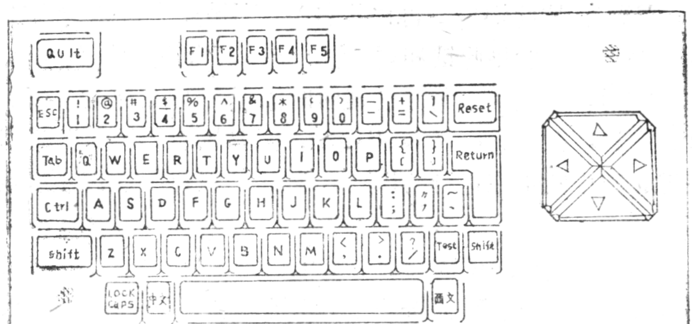


图1.2 键盘

键盘和主机通过一条26线扁平电缆连接，直接装入主机壳内。用户通过键盘向主机输送信息。此键盘具有重复功能和N键滚动功能。

所谓重复功能是指当某键按下的时间大于一定值时，则连续不断地重复发送这个键值，并被主机接受。

所谓N键滚动是指在有多个键按下时，则根据被按下键的先后次序分别检测到并输入主机，同时也显示在屏幕上。

键盘的技术性能：

键数：69个

字编码方式：ASCII码

编码数：128个

接口器件：KB3600-PRO

特殊键功能说明：

Capslock：大小写字母转换键，又称锁定键，当按下此键,再按字母键时,则字母全是大写。若又按一次锁定键，即抬起锁定键，所有再按的26个英文字母键都是小写。

Shift：当同时按此键和另一键时，产生按键上半部标注的字符输入主机并显示。

ESC：按此键后，再按其它ASCII键产生新的ASCII码输入主机，一般将此键和◁，▷，△，▽键用于屏幕编辑。

Ctrl：这是一个与其它键同时按下产生某种控制码的键，例如：在运行BASIC程序时，同时按下Ctrl和字母C两个键，则中断正在运行的BASIC程序，回到BASIC提示状态。

Tab：屏幕定位键。如屏幕上已定了11、21、31、41四个位置,光标在13那儿,按一下Tab键就移到下一个定位21。

Return：回车键，程序中通常用↲表示，用于结束一

个命令，或者是换一行，功能如象打字机上的CR键。

Reset：同时按下本键与Ctrl键，使机器系统复位（或称重置系统，即重新启动系统）。

▷：按此键使屏幕上的光标右移，并把它经过的那些符号当做您从键盘输入一样送入主机。

◁：按此键使屏幕上的光标左移，并把它经过的那些符号从主机中删掉。

△：光标上移。

▽：光标下移。

Test：当同时按下Ctrl—Reset—Test三个键时，主机进行自检。

F1：进入ASCII方式，按此键后，再按的各键主机按ASCII码接收并显示。

F2：进入拼音方式，按此键后，即可用对应的拼音字母输入汉字。

F3：进入区位码方式，按此键后，即可用区位码输入汉字。

F4：计算机系统把该键留给用户作为自定义的功能键之用，键码为：14

F5：留给用户作为自定义的功能键之用，键码为：06

Quit：按此键使系统从监控状态返回到BASIC状态，按此键不破坏内存里原有的BASIC程序。相当于同时按Ctrl-C两个键。

1.1.3 显示器

显示器是计算机最基本的输出设备。是人—机对话的主要工具之一，它不仅显示用户输入的字符，以便即时看到已

输入的字符是否正确，而且能显示主机运行的过程及结果，并且还会显示主机对用户所提出的要求和命令。

中华学习机的显示器可以用通常家用的彩色或黑白电视机（PAL—D制式）代替，也可以用单色或彩色（PAL—D制式）监视器。

对于彩色或黑白电视机，要求是PAL-D制式的，即能接收我国电视节目的电视机。

中华学习机的显示方式有五种。

1. 字符显示

能显示5×7点阵构成的96种字符。显示屏上共有24行，每行40个字符，即可显示960个字符，可以显示正相的（黑底白字），也可以显示反相的（白底黑字），或闪烁显示（即上述两种情形的交替）。

2. 低分辨图象显示

可显示48×40（1920）个彩色小方块。每个小方块可在16种不同颜色中选择。

3. 高分辨图象显示

可显示192×280个彩色点。每个点可在8种不同颜色中选择。

4. 混合显示

可以是1、2两种方式的混合，也可以是1、3两种方式的混合。各种方式的转换均由程序通过软开关实现。

5. 中文方式

中文显示采用前述高分辨图象显示方式以16×16的点阵显示构成7445种图形字符，其中包括6763个汉字及英文字母、日文假名、拉丁字母、俄文字母、希腊字母等。显示屏上共能显示11行。前10行，每行显示17个汉字，共可以显示汉

字170个。第11行用来显示当前的输入方式和相应的提示信息。

1.1.4 打印机

中华学习机可通过打印机接口板与打印机连接。打印机接口卡可插入主机扩充槽与主机50线总线相连。中华学习机可配CP—80、RX—80、MX—80、FX—80或MX—100、FX—100等多种型号的打印机。在汉字管理程序中，打印机驱动程序的设计仅适用于九针图形打印机。其型号要求必须与EPSON MX—80Ⅲ型打印机相兼容，如：FX—80Ⅲ、RX-80Ⅲ、CP—80Ⅲ、FX—100⁺和YAMATO等型号的打印机。对于MX—80Ⅱ、FX—80Ⅱ、RX—80Ⅱ可以由用户修改控制命令的方法解决，修改方法见第十三章第4节。

有了相适宜的打印机就可以把用户的文件、程序、报表、图形等打印出来，作为资料保存使用。

1.1.5 软盘和软盘驱动器

软盘的作用是保存程序、数据等资料信息，它与唱片相似。软盘驱动器是实现对软磁盘进行读、写操作的硬设备。中华学习机要求使用5.25英寸的软盘驱动器；5.25英寸单面单密度，容量为143K字节的软磁盘，见图1.3。

1. 软盘片使用注意事项

①软盘片不能受重压，不可弯曲，使用后要及时插入盘片的纸盘套内，立放于软盘盒中。

②使用时将软盘片有标签的一面朝上，用右手拇指按着有永久性标签处，将有磁头读/写窗口的一边先插入驱动器，用拇指慢慢的试探性地推入软盘片，直到盘片全部进入驱动

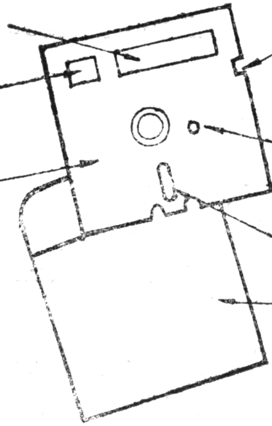


图1.3　软盘

器，再关好驱动器的门。

③取出时用食指和拇指捏着盘片的边缘，反方向抽出。

④不能在驱动器旋转和亮灯时，打开驱动器门，抽出或插入软盘片。如此容易损坏软盘片，丢失盘片上的信息。

⑤使用中要保持盘片的清洁，避免灰尘和油污的污染，绝对禁止用手触摸暴露在磁头读/写窗口和索引孔中的磁盘部分。

⑥盘片的保存温度最好在10°C—50°C之间，避开磁场的干扰，保持干燥。

2． 软盘驱动器使用注意事项

①对软盘驱动器要做到轻拿轻放，避免剧烈震动，以防定位机构松动。

②避免在软盘驱动器工作时移动盘驱本身。

③禁止在开机状态插或拔软盘驱动器和软盘驱动器接口卡。

④不可在马达转动时抽、插软盘片。

⑤温升变化范围应小于每小时15°C,环境温度应在10°C～38°C之间。

⑥避免灰尘侵袭，不要将沾有灰尘的软盘片插入到软盘驱动器中。

⑦磁头可用专用的清洗软盘片或磁带清洗液。

⑧软盘驱动器应远离磁场发生源，不用时不要将软盘片留在驱动器中。

1.1.6 录音机

录音机是一种廉价的外部存储器。由于一旦关掉电源主机内存中的内容就会消失，所以除了用软盘外，还可用录音机磁带保存程序、数据等资料信息。中华学习机要求录音机具有录放的功能并带有ERA和MIC3.5mm插孔。建议使用工厂推荐的配套盒式录音机（价格为几十元）。

1.1.7 电源

在主机壳内左边有一个开关电源。输入电压为220V、交流50Hz，最大功率小于25W。由开关电源输出±5V、±12V供主机及磁盘驱动器接口工作。

中华学习机不需要专门的稳压装置。

1.2 部件之间的线路连接

首先从中华学习机主机箱中小心取出主机，并清点随机带的视频电缆，录音机电缆，使用手册等附件。

主机右上角有一个电源指示灯（小孔内）。电源一旦接通，此灯便发亮。前半部是键盘，机壳侧面装有各种接插件

与外部设备连接。

主机上的接口请参看图1.4。

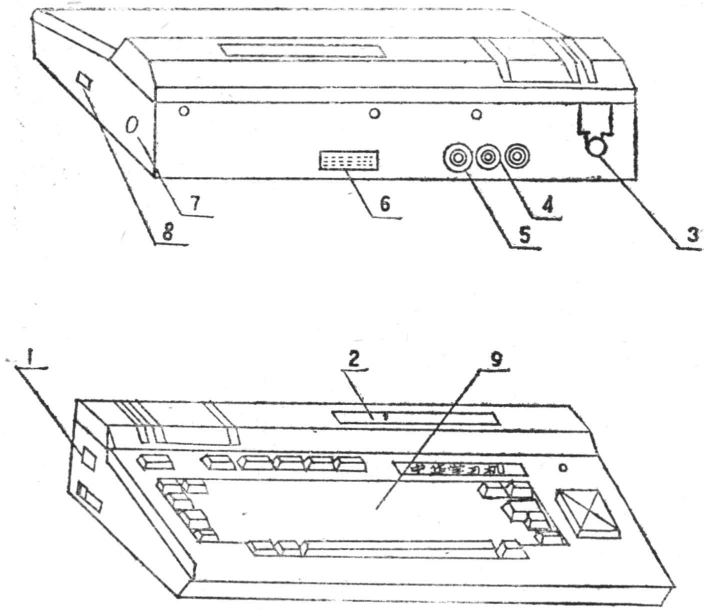


图1.4　　主机接口图

1.电源开关；2.扩充槽口；3.电源线;4.电视机接口;5.监视器接口；6.磁盘驱动器接口;7.录音机接口;8.游戏杆接口；9.键盘。

提醒用户注意：如果读者是初次安装计算机系统，各部份线路连接之后，请不要急于打开电源开关，仔细检查各部件之间的线路连接确认正确无误后，再开启电源，如有老师指导，最好请老师检查后，再开电源。

1.2.1 主机和屏幕的连接

屏幕可以用电视机（彩色或黑白）或监视器（彩色或黑

白），但连接的方法不一样。

1. 电视机连接（参看图1.5）。

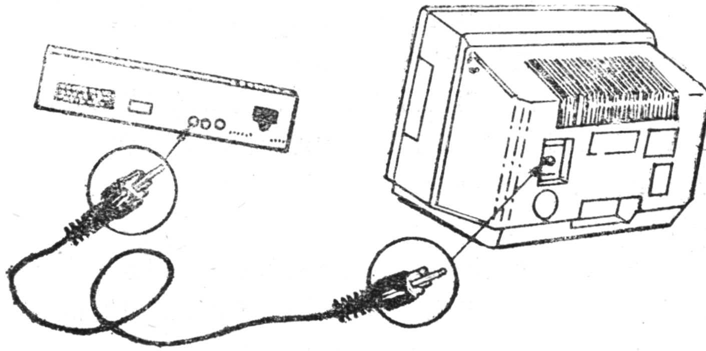


图1.5　电视机的连接

第一步：将主机和电视机放在桌上，并配备两个电源插座。

第二步：把电视机的天线插头去掉。

第三步：用视频电缆线一头插入主机的电视机接口，一头插入电视机天线插座，要求电视机天线插座的输入阻抗为75Ω。

第四步：确认主机电源开关处于“OFF”位置（见图1.6）。

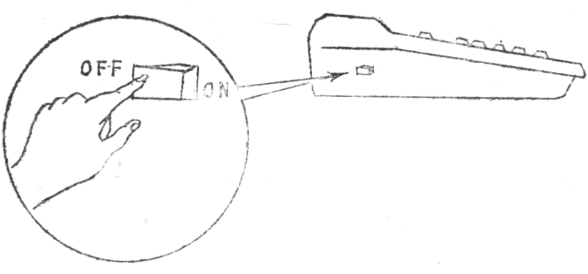


图1.6　主机电源OFF位置

第五步：将电视机电源插头和主机电源插头分别插入电源插座。

第六步：把电视机的音量开关拧到最小位置。

第七步：先开电视机再开主机。

第八步：当使用黑白电视机时，首先调准电视机频道（以出厂厂家要求为准）。调节频道微调及对比度、亮度旋纽，直到屏幕出现清晰的“ZHONG HUA XUE XI JI”字样为止。

当使用彩色电视机时，同样先选好一个频道，并通过自检方式进入显示彩条图象，然后调节频道微调及饱和度、亮度、对比度旋纽，直到屏幕上的彩条满意为止。以后可把此频道作为连接计算机专用频道，再开机时就不用调机了。

注：主机的电视机接口输出阻抗是75Ω。

2． 监视器的连接（参看图1.7）

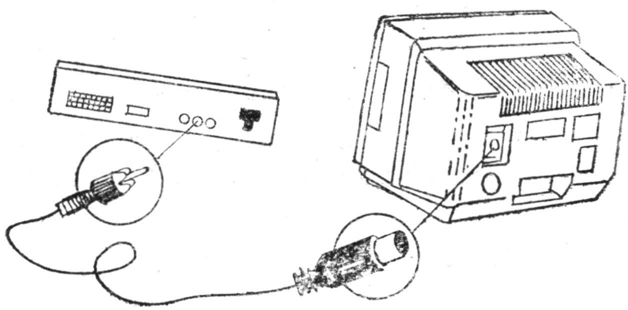


图1.7　监视器的连接

第一步：备两个电源插座，把主机和监视器放于桌上。

第二步：用视频电缆线一头插入主机的监视器接口，另一头插入监视器IN插座。

第三步：确认主机电源为“OFF”位置。

第四步：把主机电源及监视器电源的插头分别插入电源插座。

第五步：先开监视器后开主机，屏幕将出现清晰的“Z-HONG HUA XUE XI JI”字样。

1.2.2 主机和录音机的连接（参看图1.8）

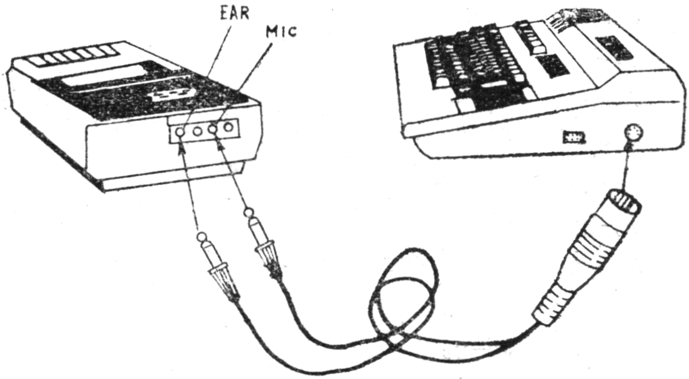


图1.8　录音机连接图

第一步：确认主机电源为“OFF”位置，再备一个电源插座（除主机、显示器电源插座外）。

第二步：把录音机电缆线的一端（五芯插头）插入主机侧面的录音机接口，必须注意：五芯插头的凹口部分在正上方。

第三步：电缆线的另一端的两个3.5mm插头，分别插入录音机EAR插孔（连接主机IN）和MIC插孔（连接主机OUT）。

第四步：把录音机电源线插入电源插座。

1.2.3 主机和扩充卡的连接

主机的右上方有个50线的扩充槽，可插多种外设卡，扩

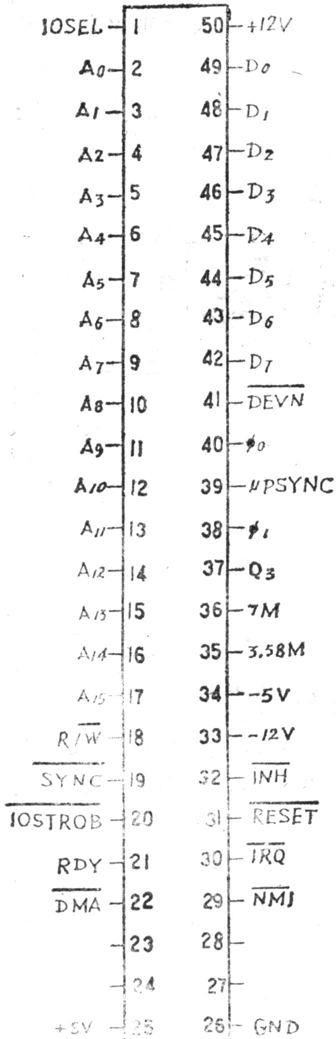


图1.9　扩充槽信号排列图

充卡（如打印机卡，音乐卡，A/D转换卡等）。扩充卡尺寸太大插不进去，可采用50线转接板把槽口抬高，再插外设卡或扩充卡。

第一步：确认主机电源为“OFF”位置。

第二步：用手轻轻压下扩充槽盖，可看到印制板上安装了一个50线插槽。

第三步：把卡按元件面朝前，焊接面朝后的方向对准槽口往下插。必须注意：一定要平插插平，以免错位。

扩充槽信号排列参看图1.9。

扩充槽的槽号，用户可以用短接帽来选择，槽号可以为1，2，4，5，7，中的任一个。3号槽已用于汉字系统，6号槽定于磁盘驱动器接口，故不可再选用它们。出厂时一般都设在第1槽，即将两个短接帽插在标有“1”的位置上。选择槽号插头位于主机板的右侧，并标有槽号。如需改槽号，需打开主机壳，把原在一槽位置的两个短接帽取下并插在你所需要的槽

号插头上。

1.2.4 主机和打印机的连接

用1.2.3节介绍的方法将主机和打印机卡连接好后。再用打印机与主机的20线（扁平）连接电缆连接主机与打印机。

第一步：确认主机及打印机电源开关为“OFF”位置

第二步：将连接打印机接口卡（小）的一端及连接打印机接口（大）的一端，分别插合在主机上的打印机卡接口及打印机的接口上（注意两端接触良好）。

第三步：先开打印机后开主机电源开关。

1.2.5 主机和磁盘驱动器的连接（参看图1.10）

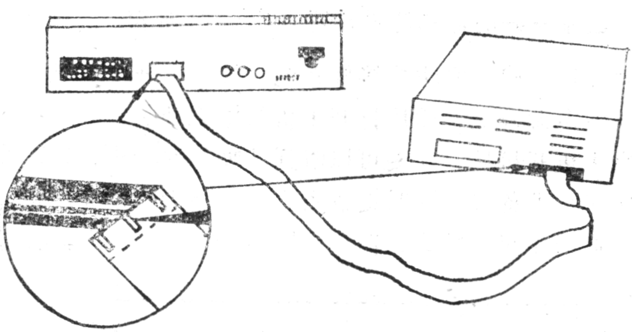


图1.10　驱动器连接

第一步：确认主机电源“OFF”位置。

第二步：把磁盘驱动器电缆线（20线扁平电源）一端插座连接驱动器，另一端插座连接主机上的驱动器接口。必须注意：插针与插孔必须一一对应，电缆线两端的凸出部分都朝上。

1.2.6 主机和游戏操纵杆的连接

九芯游戏杆插座位于主机右侧（图1.11），把连接游戏杆的插头座直接对准游戏杆接口插入，注意，两者的信号端一定要一一对应。游戏杆接口信号如图1.11。

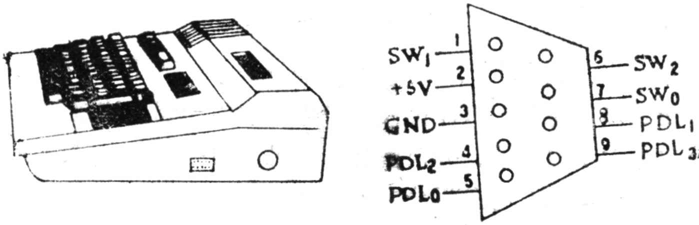


图1.11　游戏杆接口信号图

1.3 系统的启动

1.3.1 基本系统的启动与自动进入BASIC语言

首先按正确的方法连接好系统。接着打开电视机（或显示器）的电源开关，然后打开主机的电源开关。

当主机开机后，立刻可以听到“哔”的一声，这声音意味着主机已准备好，同时键盘右上方的指示灯发亮。正常情况下，此时主机自动进入西文BASIC状态，屏幕显示如图1.12。

```
ZHONG  HUA  XUE  XI  JI

VERSION  1.1

]■
```

图 1.12

显示屏上有一个被称为光标（闪烁）的小方块，它提示操作者下一个输入字符的显示位置。“]”是BASIC语言系统的提示符。

1.3.2 连接驱动器后系统的启动

当把具有软盘驱动器的系统连接好后，在开机前打开磁盘机正面中央的盖子，把DOS 3.3系统盘印字的一面朝上，水平而且轻轻地沿着磁盘机中间的细缝插入磁盘机内，当磁盘完全插入后，再关上磁盘机的小门。然后再开主机电源，显示器首先显示如图1.12，磁盘机上的指示灯发亮，且可听到马达转动的声音，直到磁盘机灯熄灭。屏幕将显示如图1.13所示的版面。

```
DOS VERSION3.3     08/25/80

APPLEII PLUS OR ROMCARD SYSTEM-
  MASTER

(LOADING INTEGER INTO LANGUAGE
  CARD)

  ]■
```

图 1.13

此时系统已在BASIC系统下。如果开机前没有将DOS 3.3软磁盘插入，系统直接进入BASIC（如1.3.1节所述），如果还需要装入DOS 3.3系统，此时可以把DOS3.3软盘插入磁盘机，并键入：

] PR# 6 ↵

屏幕将显示图1.13。

1.3.3 中西文方式的转换

当加电后，屏幕显示方式为低分辨字符方式，此时若敲“中文”键，系统进入汉字显示方式。屏幕显示方式为高分辨图形方式。屏幕可显示每行17个汉字，共10行，屏幕底部第11行为状态行。状态行左端显示“字母”两字，表示系统处于字符输入状态。当敲入F2或F3时，状态行左端显示“拼音”或“区位”，表示系统处于汉字的拼音输入方式或区位输入方式。此时按F1可回到字母输入方式。CTRL-O键可控制状态字符的显示与不显示，但不改变输入方式。

在中文方式下，敲入“西文”键，系统退出中文方式，屏幕显示回到西文低分辨字符方式。

除键盘方式（按键）进入汉字显示方式外，程序方式也可进入汉字显示方式。在BASIC程序中使用PR# 3命令可进入汉字显示方式。使用TEXT或PRINT CHR$（17）可退出汉字显示方式。

1.3.4 系统的重新启动

当加电后，系统运行正常，但由于操作失误或程序错误等原因使系统进入死循环或者使用者需要系统重新启动工作时，可同时按下Ctrl和Reset键。如此系统便重新启动并进入BASIC。这个过程亦称为系统复位。

1.3.5 LOGO语言的进入方法

中华学习机主机内除有固化的BASIC语言外，还有固化的LOGO语言，LOGO语言具有两种进入方法：

1. 在系统未装入DOS 3.3的BASIC状态下，键入：

] LG↵

即可进入LOGO语言。屏幕显示如图1.14

```
CHINESE EDUCAION  COMPUTER

          LOGO

      VERSION  1.1

         1987.6

 ?
```

图 1.14

2. 在系统已装有DOS 3.3的BASIC状态下，键入：

] MAXFILES1↵

] LG↵

即可进入LOGO语言。屏幕显示如图1.14所示的版面，LOGO语言的提示符为"？"。

1.4 中华学习机的自动检测

同时键入Ctrl-Reset-Test三个键且最后放开Test键，可进行主机检测，亦称主机自检。即对RAM、ROM进行检测，显示器显示相应的检测内容和结果。如图1.15。

其中RAM部分是测＄0000～＄BFFF48K RAM。

BNK1、BNK2部分是用两个软开关测试＄D000～＄FFF RAM。

当测试正确，屏幕显示OK，如有错，屏幕相应处显示

MEMORY — TEST

TIMES：0001

RAM	BFFF	OK
BNK1	FFFF	OK
BNK2	FFFF	OK
ROM1	BFFF	LOGO
ROM2	FFFF	CEC-BASIC
AUX1	BFFF	HZTABLE
AUX2	FFFF	HZPROGRAM
AUX3	5D99	CECWL

图 1·15

ERR。

ROM1是测试LOGO语言的。

ROM2是测试中华学习机CEC-BASIC和监控内容，ROM1和ROM2是合在一块27256ROM片上。

AUX1测试汉字码表内容。

AUX2测试汉字管理系统内容。AUXI和AUX2合在一块27256ROM片上。

AUX3测试两块1兆位的汉字库内容。

测试正确，屏幕将显示相应的测试内容名称，如果测试出不为所知的内容，屏幕将显示UNKNOWN，当没有加汉字库时，屏幕将在相应位置显示NULL。

当RAM，ROM测试完后，程序进行彩色显示测试，屏

幕将显示16种不同颜色的彩条。

自检程序可连续循环地测试，并在TIME：一栏显示测试的次数。如需调试彩电，可在屏幕显示彩条时按任何一个键使屏幕一直显示彩条，直到您调试好彩电，再按任一个键又进入自检程序循环测试。如要中断自检，请同时按下Ctrl—Reset键，程序便进入BASIC状态。

1.5 中华学习机的主要技术指标

CEC—I中华学习机的主要技术指标、性能如下：

1. 主机

①中央处理器：6502，8位微处理器。

②内存容量：64K字节RAM。

③32K字节ROM，固化监控程序及BASIC、LOGO（子集）语言。

④专用集成电路：一片门阵列器件MMU为存储器管理部件，一片门阵列器件IOU为输入/输出管理部件，一片可编程阵列逻辑电路PAL产生时序信号。

⑤汉字系统：汉字系统已做在主机板上，可根据用户需要取舍。该系统提供拼音、区位、音形等输入方式，采用全点阵的国标一、二级汉字字库（两片一兆位ROM），包括6763个汉字及外文字母等。

⑥显示器接口：一个是经过调制器输出的射频信号，可连接PAL制式彩色或黑白电视机；另一个是全电视视频信号（没有音频）输出，可接标准的PAL制式彩色或单色监视器。

⑦盒式磁带机接口：该接口可与一般家用录音机连接，作为廉价外存设备。提供输出电压是25mv，输出阻抗100Ω，要求输入信号峰—峰值电压是1V，输入阻抗12KΩ。

⑧磁盘（软盘）驱动器接口：磁盘驱动器接口已做在主机板上，用于连接一个5.25英寸磁盘驱动器。

⑨游戏控制杆接口：九针的游戏杆插座提供三个TTL电平开关量输入，四个模拟量输入。

⑩扬声器接口：内接0.25W、8Ω扬声器。

⑪键盘：键盘是标准的打字机键盘，具有69个键，包括大小写字母、数字及一些特殊功能控制键。采用弹簧键体，键盘通过26线扁平电缆直接连接主机板的键盘接口，装在主机内。

⑫扩充槽口：在主机板上留有一个与APPLE Ⅱ系列兼容的50线输入/输出插座。此插座可插打印机接口电路或其它扩充电路板。

⑬电源：采用开关电源，提供四路电源，额定负载为：＋5V、1.7A；－5V、0.1A；＋12V、0.8A；－12V、0.1A。总功耗约25W。

2. 显示器

可用彩色或黑白电视机，也可用彩色或单色监视器，其屏幕显示功能为：

①西文方式

字符构成：5×7点阵　　每帧字符：40字符×24行

显示方式：正常，反相，闪烁。

②中文方式

字符构成：16×16点阵（汉字）　　8×16点阵（AS-

CII码）

每帧字符：17字符（汉字）×10行　　34字符（ASCII码）×10行

另加一个状态行

③图形方式

低分辨率彩色图形显示：分辨率为40H×48V16种颜色

高分辨率彩色图形显示：分辨率为280H×192V　　8种颜色

3. **软盘驱动器**

5.25英寸软盘驱动器，采用单面单密度记录方式，格式化后容量为143K字节。

4. **盒式磁带机**

任选专用磁带机或普通盒式录音机，用普通标准录音磁带。

5. **打印机**

打印机与可扩充的打印机接口电路板相连。Centronic标准接口可配多种点阵打印机和彩色绘图打印机。

6. **系统软件**

包括监控程序、DOS 3.3、CEC—BASIC、LOGO（子集）、汉字管理软件等。

习　题

请实际安装一台中华学习机CEC—I，并熟悉各物理线路之间的连接方法。

第二章　BASIC语言的基本概念

本章针对中华学习机CEC-Ⅰ，介绍BASIC语言的一些基本概念。这些概念不仅适用于中华学习机，而且也适用于一般机器的BASIC语言。掌握了BASIC语言后再去学习其它高级语言也就不困难了。

2.1　什么叫BASIC语言？

2.1.1　BASIC是什么意思？

对于BASIC语言，读者可能并不陌生。因为很多场合都能碰到它。但究竟是什么意思，有些读者，特别是计算机技术的初学者可能就不太清楚了。单就BASIC这几个字符来说，它是英文Beginer's All-purpose Symbolic Instruction code的缩写，意思是初学者的通用符号指令代码。它代表了一种语言。但这种语言不同于日常生活中用于交流思想感情的自然语言，它是一种用于描述小型数值计算问题和事务管理的计算机算法语言。

就一般来说，BASIC语言总是与某类微电脑联系在一起。如第一章介绍的中华学习机CEC-Ⅰ，它使用的BASIC语言通常说成CEC-BASIC，表示BASIC语言适用于具体的CEC-Ⅰ机。类似的有APPLEⅡ BASIC，紫金Ⅱ BASIC，IBM PC BASIC等等。

对于初学者来说，只要掌握了一台具体机器上BASIC

语言的使用方法，其它机器上的 BASIC 语言也就容易触类旁通了。因为从总体语法规则来说，不同类型机器上的BASIC语言总是大同小异。因此，建议读者在学习一种语言（不论是什么语言）时，先结合一台具体机器的语言，边学边用，学用结合，这样可以达到多快好省的效果。

2.1.2 BASIC语言的特点

BASIC语言自身有很多特点，但归纳起来，最重要的特点有：

1. 语法规则比较简单、直观，容易为大多数人理解和接受。因而容易学会使用。

2. 程序结构简便。由于大多数微机上的 BASIC 语言都采用解释性的方法来实现，因而对 BASIC 语句的增删改都比较容易。

3. BASIC 语言具有很强的人机会话功能。人们在使用语言时，能够随时掌握语言的执行情况，能够及时调整和交换信息。

由于 BASIC 语言具有简便实用的特点，因而目前几乎所有的微机上都配备了 BASIC 语言系统。随着计算机技术的不断发展，BASIC 语言自身也在不断改进和提高。总的发展趋势是：功能越来越强，应用越来越方便。我们着重介绍的中华学习机上的 BASIC 语言，已经固化在主机的只读存贮器中，称为CEC-I BASIC，简称BASIC，其功能相当于扩展BASIC。它除了具有APPLESOFT BASIC的功能外，还提供了对汉字的输入、显示和打印等增强功能。可以说，CEC-BASIC是一个小型的多功能语言系统。

2.1.3 BASIC语言的使用方式

BASIC 语 言有两种使用方式（亦称执行方式）。一是人机对话方式。所谓人机对话方式，就是指在机器上敲入一个命令后机器就立即执行之，其执行结果马上在屏幕上显示出来，这种一问一答的方式，最适宜于初学者作为熟悉BASIC 语言练习之用。因为看得见，很实在。另一种方式就是程序设计方式。这种方式的特点是将需要进行的计算或事务管理工作编制为程序，再输入机内一起执行。这种方式速度快，程序能够自动执行。适宜于实际应用。当然，两种方式在使用中并不能截然分开。在本书中，我们着重介绍程序设计的工作方式。

下面用一个例子来说明两种执行方式的主要区别：

例2.1 计算梯形的面积。公式为S=（上底+下底）×高/2

假设用A表示上底，B表示下底，H 表示 高。给出一组数据为 3 ， 4 ， 5 （即上底＝3 ，下底＝4 ，高＝5 ）。

用对话方式，BASIC执行情况如下：

```
]A=3        ↵          将上底值 3 送到A
]B=4        ↵          下底值 4 送到B
]H=5        ↵          高的值 5 送到H
]S=(A+B)*H/2 ↵           计算梯形面积
]PRINT S↵               打印计算结果
17.5                   运算的结果
```

用程序设计方式如下：

```
]10 INPUT  A, B, H  ↵     使A，B，H 得到值
]20 S=(A+B)*H/2 ↵         计算面积
```

] 30 PRINT S↵	打印计算结果
] RUN ↵	运行程序
? 3，4，5 ↵	输入数据
17.5	输出运算结果
]	回到BASIC状态

如果读者亲自上机一试，就会发觉：对话方式的特点是敲入一个语句就立刻执行那个语句，并且语句之前不带行号（即每行前面给出的正整数）。而程序设计方式是把所有语句全部输入机内，最后发布运行命令RUN，机器才从行号小的语句开始执行，输出运行结果。如果对上述语句中的符号不清楚，没关系，待下一章介绍后就自然明白了。

2.1.4 本书用到的书写记号约定

为了叙述的方便，本书经常用到以下一些记号。这些记号不是 BASIC 语言自身的内容，引入这些记号纯粹是为了说起来简便。

[…]——表示方括号中的内容可根据实际需要进行取舍，称为待选项。

<…>——表示尖括号中的项在实际书写程序时必须填入具体的内容，但程序中并不书写尖括号。

↵——表示回车键，即键盘上的 RETURN 键。

Ctrl — Reset ——表示先按住 Ctrl 键不松手，再敲 Reset 键。

2.2 BASIC语言用到的符号

任何一种语言，包括自然语言在内，都要用到一些符号，如汉语中使用的符号除了汉字本身外，还有标点符号，括号等等。英语中用到的符号有26个大小写字母，标点符号，数字符号等等。那么BASIC语言中需要哪些符号呢？

2.2.1 字符集

所谓BASIC语言的字符集，就是BASIC语言中可能用到的所有符号。根据符号的性质，可以分为以下几类：

1．字母

BASIC语言中要用到的字母有：A，B，C，D，E，F，G，H，I，J，K，L，M，N，O，P，Q，R，S，T，U，V，W，X，Y，Z。字母必须用大写。

按下键盘上的 CAPLOCK 键可以控制字母为大写。

2．汉字

中文方式下的BASIC语言可用全点阵的一二级汉字库中的6763个汉字。

3．数字

BASIC中可用的数字符号有：0，1，2，3，4，5，6，7，8，9共十个数字符号。

4．其它符号

BASIC语言中常用的其它符号有：＋－＊／∧（）＝ < >，$ % · ： ； " ? 等等。

BASIC语言能够使用的大多数符号都列在键盘上，请

参看CEC-I键盘。

2.2.2 BASIC语言的保留字

在BASIC语言中，为了描述问题的需要，某些字符已经结合在一起，形成一个单词如INPUT，FOR，TO，DATA 等等，这些单词在语言中已有固定的含义并保留作专门之用，称为保留字。顾名思义，保留字就是保留给系统专用的单词，用户不得再派用场。BASIC 语言的保留字如下：

&

ABS	AND	ASC	AT	ATN
CALL	CHR $	CLEAR	COLOR=	CONT
	COS			
DATA	DEF	DEL	DIM	DRAW
END	EXP			
FLASH	FN	FOR	FRE	
GET	GOSUB	GOTO	GR	
HCOLOR=	HGR	HGR2	HIMEM:	HLIN
	HOME	HPLOT	HTAB	
IF	IN#	INPUT	INT	
	INVERSE			
LEFT $	LEN	LET	LIST	LOAD
	LOG	LOMEM:	LG	
MID $	MUSIC			
NEW	NEXT	NORMAL	NOT	
	NOTRACE			
ON	ONERR	OR		
PDL	PEEK	PLOT	POKE	POP

	POS	PRINT	PR#	PLAY
READ	RECALL	REM	RESTORE	
	RESUME			
	RETURN	RIGHT$		
	RND	ROT≠	RUN	
SAVE	SCALE=	SCRNC	SGN	
	SHLOAD			
	SIN	SPC(		
	SPEED=	SQR	STEP	STOP
	STORE	STR$		
TAB(	TAN	TEXT	THEN	TO
	TRACE			
USR				
VAL	VLIN	VTAB		
WAIT				
XPLOT	XDRAW			

2.3 常量、变量和字符串

2.3.1 常量及其写法

什么叫常量？所谓常量（或称常数）就是其值是固定不能变化的量。例如：数值195，一经写定其值就不能再变成352了。根据常量的书写形式，常量可分为数值常量、字符串常量和逻辑常量。

1. 数值常量

所谓数值常量就是其值是固定不变的且组成符号仅由数字、正号（＋）、负号（－）、小数点（·）和指数符号E

按一定规则组成。

例2.2　+18.73，−39.256，1988，1.285E+3（E代表底数10，即为$1.285_{10}{}^{3}$）。

这些数都是正确的数值常数。数值常数的书写除了指数必须用E来表示底数10以外，原则上同数学中一样。在书写数值常量时必须注意+−·E符号的正确位置。数值常量的值就是它所表示的代数值。

2．字符串常量

字符串常量是用引号"括起来的字符序列。

例2.3　"我是一个学生"，"ax+by"，"abcdef"，"this is a…"，"53+24=77"，"w（5，3）"，"1988"

以上均是正确的字符串常数。在书写字符串常量时，引号"必须成对使用。如"ax+b和ax+b"都不是正确的字符串常数，因为它们各自都只有一个引号。字符串常量的值是指两个引号之间的字符（包括字符间可能有的空格）。如"He is a student"字符串的值是He is a student。

3．逻辑常量

BASIC语言中还有一种量，它专门用于描述某种条件是否成立。这种量称为逻辑常量，它用于代表真或假。通常用1代表真，用0代表假。

2.3.2 变量

在数学中我们常常用X，Y等字母来表示一些未知的量。这种未知量其值是可以变化的。例如X，在某种情况下其值可以是3.125，在另一种情况下它又可以是193.24。类似X，Y的这种未知量，在BASIC语言中称为变量。根据变

量的取值情况，变量又可以分为数值型变量、字符串变量和逻辑型变量。数值型变量只能取数值为其值，字符串变量只能取字符串为其值，逻辑型变量只能取真或假值为其值。

在 BASIC 语言中，如何表示常数和变量呢？表示常量很直观，只需把常量写在需要的地方就行了。而要表示变量就必须先给变量取个名字，我们称为变量名。变量名是用字母开头的字母或数字的序列。例如前面的 A，B，H 就是三个不同的变量名。

例2.4　X，Y，X_1，Y_5，AB，H，S，T_1

这些都是正确的变量名。而A－5，X）3.5，T＋5都不是正确的变量名。在给变量取名时必须注意以下几点：

1．变量名必须用字母开头，其后可以紧跟一个字母或一个数字符号。如果一个变量名超过了两个字符，那么只有前面两个字符有意义，如ABD和ABCDE，均把 AB作为变量名，因而ABD和ABCDE实际上是同一个变量名。

2．变量名的组成符号除了英文字母和数字符号以外，不能夹杂其它字符。初学者容易忽视的是常常在变量名中夹杂空格，这是不正确的。

3．BASIC保留字不能再用作变量名。

只要遵守以上规则，对变量的取名就是自由的了。建议在给变量取名时最好有一定的含义。如时间用T，高度用H，面积用S等等，使人们一目了然。

2.3.3　数据类型

根据变量取值的范围，变量又可以分为三种：实型变量、整型变量和字符型变量（或称串变量）。实型变量取值为实数即带有小数或指数形式的数值。整型变量取值只能是

整数，即仅由数字、＋、－符号组成的数。串型变量只能取值字符串。

1. 数据类型的标志

对于一个变量名，为了明确表示其数据类型，在构造变量名时就给出一定的标志。如果一个变量名之后紧跟一个%符号，则表示该变量是整型变量。如A%，X%等。如变量名之后紧跟的是$标志，则意味着该变量是串型变量。一个实型变量就是变量名本身，即变量名之后不跟任何标志。

一个变量究竟应该属于什么类型，实际使用中不难划分。总的原则是：如果一个变量要参与算术的加减乘除以及乘方运算，那么肯定不能是串型量，多数情况下还是取实型量为好，若取整型的话，运算结果的小数部份有可能丢掉。

2. 数值的表示范围

对于一台具体的机器来说，表示数值的大小总是有限的。但不同类型的机器能够表示的数值范围的大小亦不相同。对于中华学习机CEC-Ⅰ来说，其表示数值的范围如下：

|实数|$\leqslant 10^{38}$，如果实际数值的绝对值大于10^{38}，则会装不下，这种情况称为上溢。如果数值的绝对值小于10^{-39}，则当成溢出处理，这种情况称为下溢。

|整数|$\leqslant 32767$，整数取值只能在－32767和＋32767之间。

串型量取值至多255个字符。当然可以一个字符也没有，这种串型量称为空串。

为清楚起见，请看下表。

2.3.4 数值的输出格式

在CEC-BASIC中，系统根据数据的取值情况，自动

数据类型	标　志	例　子	取值范围
实型量	无	A	$\|A\| \leqslant 10^{38}$
整型量	%	B%	|B%|≤32767
串型量	$	C$	0～255个字符

确定数值数据的输出格式，其原则是：

1．如果输出的是一个负数，则在数据前面带一个负号"－"。如果是正数，则输出数据不带任何符号（即是一个无符号数，它表示一个正数）。

2．若0≤|数值|≤999 999 999且为整数，则作为整数原样输出。

3．若0.01≤|数值|≤999 999 999.2且为实数，则按实数原样输出。

4．若数值不属于2，3种情况，则按指数形式（又称科学计数法）输出。输出格式为：

SX.XXXXXXXXEVTT

其中：S是数的符号（正号或负号），可以是＋或－，X为0～9之间的数字，·为小数点，E为底数10，V是指数的符号（＋或－），TT为指数值。

例2.5　若A＝0.000009，输出格式为9E－6

若B＝123456789999999，输出为1.23456789E＋15

需要特别注意的是：

①BASIC输出数据时至多只能达到9位有效数字，但输入数据时可以超过9位数字。

②BASIC对计算结果的舍入原则是：当不足9位时给

出精确值，达到或超过9位时给出9位数的近似值，在第10位上进行四舍五入。

③当一个数被转换成整型数时，则取小于或等于该数的最大整数作为转换后的整数结果。因此，0.999转换成整型数是0，而－0.05转换成整数则是－1。

2.4 函数

为使实际应用方便起见，BASIC把诸如数学中的三角函数，自然对数，开平方，求绝对值等运算用专门的符号表示出来，当用户需要时就在程序中直接调用。根据函数的用途可分为数学函数，字符串处理函数、类型转换函数等等，详细情况请参见第七章有关节段。函数的一般形式如下：

<函数名>（<形式参数>），其中函数名是BASIC语言给标准函数取的名字，形式参数是调用时必须提供的参数。由于这里仅起形式上的作用，因而称为形参，有些资料又称哑元。调用时实际提供的参数才起真正的作用，因而称为实在参数（简称实参），又称实元。下面列出常用的几个函数：

SIN（X）—求SINx的值。

COS（X）—求COSx的值。

ABS（X）—求x的绝对值。

SQR（X）—求正数X的平方根（$\sqrt{x}$）。

EXP（X）—求e^x的值。

LOG（X）—求X的自然对数值。

INT（X）—取X的整数部份值。

实际应用中要注意：

①凡是能够用函数来表示的运算最好用函数运算。

②所有三角函数的参数都用弧度表示。如果提供的是角度，那么必须转换成弧度。

③X可以是表达式（参见下一节）。

④圆括号不能少掉。

⑤函数运算后一定能够得到一个值。

2.5 表达式

我们已经讨论了常量、变量和函数。如果把这些量用相应的运算符按一定的规则联系起来就构成 BASIC 语言中所谓的表达式。表达式可以分为四种：算术表达式，关系表达式，逻辑表达式和字符串表达式。

2.5.1 运算符及其优先级别

1. 算术运算符

BASIC中要用到的算术运算符有：

+	-	*	/	∧
加	减	乘	除	乘方

算术运算符可以用来构成算术表达式以进行算术运算，其运算规则与数学中一样。首先进行乘方，其次乘除，最后进行加减运算。如果有圆括号，那么括号优先。同级运算从左到右依次进行。算术运算的结果必定是一个数值。

2. 关系运算符

<	<=或=<	=	>	>=或=>	<> 或 ><
小于	小于或等于	等于	大于	大于或等于	不等于

关系运算符用以对两个算术项进行比较，结果产生一个逻辑值真或者假。CEC-BASIC用1代表真，用0代表假。由于BASIC中不允许把几个关系运算符连在一起，因而不存在优先级关系，如a>b>c=d的写法是不正确的，它必须换成逻辑表达式来书写。

3. 逻辑运算符

BASIC的逻辑运算符有：

NOT	AND	OR
非	与	或

逻辑运算符用于进行逻辑运算，运算结果仍然是一个逻辑值真或假，运算规则如下表所示。假设A、B是取值为真（1）或假（0）的逻辑变量，A和B的逻辑运算结果也示于表中。其中，1代表真值，0代表假值。

A	B	NOT A	A AND B	A OR B
1	1	0	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	0

从上表的运算结果可以看出逻辑运算符的运算规则有如下的规律：

NOT A是对A的否定。如果A为真，那么NOT A的结果是假，如果A为假，NOT A的结果为真。

对于AND运算，只有当AND的两端同时为真（即A为真B为真）时，A AND B的结果才为真，其余情况均为

假。

对于OR运算，只有当OR两端的值同时为假时，A OR B的值才为假，其余情况均为真。

逻辑运算符的优先级别是：NOT AND OR。

2.5.2 算术表达式

1. 什么叫算术表达式？

算术表达式是指一个数值常量、数值型变量或算术函数以及用算术运算符＋－＊／∧和圆括号（）将这些量按一定规则连接而成的式子，我们都可以称之为算术表达式。一个数值常量或数值变量或算术函数是算术表达式的特殊情况。为了区分，有时将它们称为简单算术表达式，而把带有运算符的表达式称为复杂算术表达式。在不致混淆的情况下一律称为算术表达式。

例2.6 1988，－3.72，1.534E+10，X，Y1，A%，SIN（3）这些都是简单算术表达式的例子。

例2.7 B*B－4*A*C，（1+SIN(X))∧2/(Y*Y+SQR(1+2*Y+5*Y∧5))，这是两个复杂算术表达式。从理论上来说，利用算术运算符和圆括号，我们可以把一个算术表达式构造得非常复杂。

2. 算术表达式的书写规则

利用 BASIC 语言来进行数值计算，必须将数学式子写成 BASIC 所允许的算术表达式的形式。其书写原则是：

①把乘号×写成＊，并且不能省写或简写。除号÷写成／。

例2.8 将下列数学式写成算术表达式

$$\frac{(A+B)\sqrt{X^2+1}}{C+D} \Rightarrow \text{(A+B)*SQR(X*X+1)/(C+D)}$$

$$\frac{(A+B)H}{2} \Rightarrow \text{(A+B)*H/2}$$

②注意成对使用圆括号，不论需要用到多少重括号，算术表达式中只能使用圆括号（ ）。

例2.9　将下列数学式写成算术表达式。

$$\left(\frac{A+B}{X+Y}\right)^3 \Rightarrow \text{((A+B)/(X+Y))}\wedge 3$$

③数学式中用到的三角函数、自然对数、e^x指数函数、开平方根等运算，都必须换成BASIC的相应函数。

例2.10　将下列数学式写成算术表达式。

$$\left|\frac{\text{Sin } e^{ax}+\text{Ln}bx}{-a\sqrt{x+a}+1}\right| \Rightarrow \text{ABS((SIN(EXP(A*X))+LOG(B*X))/(-A*SQR(X+A)+1))}$$

④区分容易混淆的字符。如字母I和数字1，字母O和数字0等。

3．　算术表达式的运算规则

算术表达式一经确定，就可以按给定的规则进行运算以得出一个具体的数值。BASIC的算术表达式运算规则同数学中一样：括号优先，多重括号时由内到外。乘方、乘除、最后是加减运算。同级运算按从左到右原则处理。

4．　算术表达式的值

算术表达式的值就是将变量值代入算术表达式以后按给定运算规则运算而得到的代数值。这个值必定是一个数值，它的数据类型一般来说是实型。

2.5.3 关系表达式

关系表达式是用关系运算符将两个算术表达式或字符串表达式连接而成的式子。其一般形式如下：

<算术表达式><关系运算符><算术表达式>或

<串表达式><关系运算符><串表达式>

例2.11 B*B－4*A*C＞0， X＞Y，H＜=1.72，"NO"＝"YES"这些都是正确的关系表达式的例子。而A＞B＞C却不是正确的关系表达式。它实质上应该写成逻辑表达式的形式。

在关系表达式中，首先进行算术运算，得到算术表达式的结果后再进行比较，最后得出一个逻辑值。在例2.11中，假设A＝1，B＝2，C＝3，那么将A，B，C的值代入关系表达式B*B－4*A*C＞0中得：2*2－4*1*3＞0，显然其值为假（0）。这就是说，在特定条件下（A＝1，B=2，C＝3），B*B－4*A*C＞0是不成立的。关系表达式通常用于构造某种选择条件。

在所有关系运算符中，对等号＝的使用要特别小心。有时我们认为不等的数（如10^{-39}和0），由于机内转换或其它处理等原因而造成两个数的相等，导致条件的满足或不满足，致使发生我们难以料及的错误。反之亦然。

2.5.4 逻辑表达式

一个关系表达式或用逻辑运算符 NOT AND OR 将关系表达式按一定规则连接而成的式子都称为逻辑表达式。

例2.12 A＞B AND C<D，NOT X=Y AND B*B－4*A*C＞0 这些都是正确的逻辑表达式。关系表达式仅是

逻辑表达式的特殊情况。对于数学中的 A>B>C 的写法，写成逻辑表达式应该是A>B AND B>C。

在一个比较复杂的逻辑表达式中，可能同时包含多种运算符，其运算优先级是这样规定的：首先进行算术运算，其次进行关系运算，最后进行逻辑运算。在各自的运算中又按各自规定的优先级进行运算。逻辑表达式最终一定能够得出一个逻辑值。

例2.13 求下列逻辑表达式的值（设A＝1，B＝2，C＝3）。（～～～～符号表示优先计算的表达式以及运算后的值）

B*B－4*A*C>0 AND（NOT 5>A）OR(A+B)*(C－B)< >B∧3 AND A+B>A*B

⇒2*2－4*1*3>0 AND（NOT 5>1）OR (1+2)*(3－2)

－8 3

< > 2∧3 AND 1+2>1*2

8 3 2

⇒－8>0 AND（NOT 5>1）OR 3< >8 AND 3>2

0 1 1 1

⇒0 AND NOT 1 OR 1 AND 1

0

⇒0 AND 0 OR 1 AND 1

0 1

⇒0 OR 1

1

⇒1

按照给定的运算规则运算后其结果为逻辑值真（1）。

2.5.5 字符串表达式

字符串表达式比较简单，它只需用串运算符“+”将两个字符串（或串变量）或串函数连接起来，从而生成一个新的字符串。

例2.14 将字符串“ABCD”和“XYZ”连接起来。

“ABCD”+“XYZ”生成一个新串“ABCDXYZ”。

2.6 BASIC程序

前面比较详细的介绍了 BASIC 语言中经常涉及到的常量、变量、字符串、函数和表达式等基本概念。下面再来说明什么是BASIC程序。所谓BASIC程序，就是用BASIC语句书写的完成某种计算或事务管理的详细步骤，请看下面的例子。

例2.15 求一元二次方程$ax^2+bx+c=0$的根。

由数学知识我们知道，求解一元二次方程根的公式为：

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$，判别式b^2-4ac如下：

$$\text{若}b^2-4ac\begin{cases}>0\text{，则有两个不同实根。}\\=0\text{，则有两个相同实根。}\\<0\text{，则有共轭虚根。}\end{cases}$$

根据一元二次方程的求解方法，编成如下的 BASIC 程序：

```
10  INPUT  A, B, C ↲
20  D=B*B-4*A*C ↲
```

```
30 IF D>0 THEN 80 ↵
40 IF D=0 THEN 110 ↵
50 X1=-B/(2*A) ↵
60 X2=SQR(-D)/(2*A) ↵
65 PRINT "共轭虚根" ↵
70 GOTO 130 ↵
80 X1=(-B+SQR(D))/(2*A) ↵
90 X2=(-B-SQR(D))/(2*A) ↵
95 PRINT "两个不等实根" ↵
100 GOTO 130 ↵
110 X1=-B/(2*A) ↵
120 X2=X1 ↵
125 PRINT "两个相等实根" ↵
130 PRINT "X1="; X1, "X2"; X2 ↵
140 END ↵
]RUN ↵
? 5, 6, 4 ↵
共轭虚根
X1=-.6                X2=.663324958
```

从上例可以看出，一个 BASIC 程序包括了这样几个部份：

1. 语句行号

这是每个语句的开始符号。它由至多 5 位 0～63999之间的无符号整数组成。语句行号数值的小和大，就代表了语句在程序中的前与后，也表示语句执行的先与后。一般来说，BASIC 程序从行号最小的语句开始依次执行。前后两个行号并不要求连续。

2. 语句

语句是 BASIC 用以完成操作动作的一些有专门意义的字符序列。如INPUT表示输入，PRINT表示打印输出。语句必须按照 BASIC 语言的语法规则来书写，而书写语句的顺序要按完成具体任务的要求依次书写（体现在语句行号的小和大上面）。在书写语句时，可以一个语句占用一个书写行，也可以几个语句写在同一行上。如110和120两句，可以写在一个行上：

110 $X_1=-B/(2*A):X_2=X_1$ ↵

3. 回车键↵

如果要把程序输入到机内，那么每个语句行输入完后必须敲一个回车键 [RETURN] 。为写起来方便，我们用↵表示回车键。回车键是一个语句行完结的标志。在书写程序时可以不写↵，但输入程序时必须敲↵键。

4. 程序结束标志END

在一个程序中，程序的结束标志 END 可写也可不写，这不影响程序的正确性（但用子程序时有例外）。严格说来，END不是BASIC的语句，它是操作系统提供的实用命令。

5. 源程序

源程序就是用BASIC语言书写的程序，亦称 BASIC 源程序，或称源程序，通常简称为程序。从以上的例子我们可以得出如下的结论：

一个程序由若干个语句组成；一个语句又由若干单词组成（这些单词包括保留字、变量、常数、函数或表达式），一个单词又由若干字符组成，这同自然语言一样，字符组成

字，字组成句，句组成文。一个程序就相当于是用 BASIC 语言写的一篇文章。我们研究 BASIC 语言，实际就是讨论由字组成句，由句组成文的规则。

在编写和输入BASIC程序时必须注意：

①一个语句总是以行号开始，以↵作为结束。

②语句行号之间最好留有一定的间隔数，以利于插入新行。如10，20，30行号比10，11，12，13行号取得好。

③如果一个语句行号上需要写多个语句，那么语句间必须用冒号"："隔开。

④一个语句行连同字符和空格不能超过239个字符。如果一个语句可能超过239个字符，那么可以拆成多个语句。

⑤用于构成变量名、保留字和命令中的字符通常用大写。

⑥BASIC 程序的执行总是从行号数值最小的行号开始依次执行（除非有转向）。因而程序中的行号不能随便乱写。

2.7 BASIC 程序的输入

输入BASIC程序之前必须事先进入BASIC系统。

2.7.1 如何进入BASIC系统

第一章已经对如何启动系统给出了详细介绍。这里仅就输入程序前如何进入系统给出简要说明。

1. 固化BASIC系统的启动

请先按第一章介绍的方法连接好中华学习机的硬件系统。打开显示器电源，再打开主机开关。屏幕显示：

```
ZHONG HUA XUE XI JI

        VERSION 1.1



]■
```

图2.1　系统进入BASIC状态

这时主机自动进入西文BASIC状态。“]”是BASIC系统的提示符，光标紧跟其后闪动，表示机器已作好准备接收输入字符。

2. 用磁盘操作系统来启动

前面介绍的是直接在主机上启动固化BASIC语言。如果我们要用磁盘上的操作系统DOS3.3来启动，那么可按如下步骤操作：

①打开磁盘驱动器中央的盖子，把DOS3.3的软盘带有标签的一面朝上，试探性地轻轻插入到驱动器中，然后关好驱动器的小门。

②打开显示器和主机电源，稍候，屏幕显示：

```
DOS VERSION 3.3  08/25/80
APPLE Ⅱ PLUS OR ROMCARD SYSTEM
  MASTER
(LOADING INTEGER INTO LANGUAGE
  CARD)
]■
```

图2.2　用DOS3.3进入BASIC状态

此时系统进入 BASIC 状态。“]”表示可以输入字符了。以上两种启动方式称为冷启动。

③如果开机之前没有把DOS3.3软盘插入到驱动器中，那么系统直接从主机进入 BASIC 状态。如果这时需要装入DOS 3.3操作系统，那么把DOS 3.3软盘插入驱动器后，在“]”提示符后敲入：

] PR# 6 ↵，这时仍然可以进入BASIC 状态，屏幕显示如图2.2所示。

3. 如果主机电源已经打开，可以用敲 Ctrl - Reset 键（先按住 Ctrl 键再敲 Reset 键）的办法启动系统。这通常称为热启动，也叫系统复位。

4. 中西文方式的转换

系统启动以后便自动进入西文使用方式。如果我们希望进入中文方式，那么只需敲键盘上的 中文 键，屏幕显示：

```
◇◇◇◇◇◇◇◇◇◇◇◇◇
◇      中华学习机        ◇
◇      CEC-I            ◇
◇                       ◇
◇ ]                     ◇
◇ 字母                   ◇
◇                       ◇
◇◇◇◇◇◇◇◇◇◇◇◇◇
```

这时便处于中文使用方式下的字母输入状态。如果要在中文方式下输入字母或其它非汉字的字符，那么就必须处于“字母”状态下。如果需要输入汉字，就必须转到相应的输入方式。下面三个键经常用到：

F_2 键——允许用汉语拼音方式输入汉字。

[F3] 键——允许用区位码方式输入汉字。

[F1] 键——允许在中文方式下输入字母或字符。

如果需要转到西文使用方式，只需敲键盘上的 [西文] 键，便进入西文BASIC状态。

需要特别强调的是：同一个程序要么都在中文方式下输入，要么都在西文方式下输入，不能一段程序在中文方式下输入，另一段程序又在西文方式下输入。中西文方式的转换将影响到输入程序的连续性。也就是说，中西文方式的转换将会把同一个程序分割成两个程序。

2.7.2 如何输入程序

1. 西文字符的输入

如果我们仔细观察键盘就会发现，有的键上只标有一个字符，如26个字母键就是如此。而有的键上却标有两个字符，如10个数字键，除了标有数字以外，还标有其它字符。如 [*/8] 键，上部标的是*，下部标的却是8。那么敲该键时如何区分呢？系统是这样规定的：如果待输入的符号所在的键上只有一个符号或是键的下部符号，那么就直接敲键。如要输入AB32，就直接依次敲 [A] [B] [#/3] [@/2] 四个键，屏幕显示的就是AB32。这种情况称为见码敲键。如果待输入的字符是键的上部字符，那么在敲该字符键时必须先按住换档键 [Shift]，然后再敲上部字符所在键。如要输

入字符*，就必须先按住 [Shift] 键，然后再敲 [*/8] 键。这种情况称为换档敲键。记成 [Shift]—[*/8]。凡是需要输入上部字符时都必须用换档敲键法。输入单个字符或下部字符时一律用见码敲键法。为便于读者练习，我们以下列程序的输入为例：

例：
```
10  INPUT A, B, H ↵
20  S=(A+B)*H/2 ↵
30  PRINT S ↵
```

敲键动作如下（[　　] 表示空格键）：

] [!/1] [)/0] [　　] [I] [N] [P] [U] [T]

[　　] [A] [</,] [B] [</,] [H] [RETURN]

] [@/2] [)/0] [　　] [S] [+/=] [Shift]—[(/9]

[A] [shift]—[+/=] [B] [Shift]—[)/0] [Shift]—

[*/8] [H] [?//] [@/2] [RETURN]

] [#/3] [)/0] [　　] [P] [R] [I] [N] [T]

] S

其中每行开头出现的“]”是系统提示符，我们不能敲入。

2．汉字的输入

中华学习机上目前提供了两种输入汉字的方法。一是汉语拼音，二是区位码。这里简介拼音输入法，详细情况请参见第十三章。

我们以输入下列程序为例：

```
10  PRINT "中华学习机" ↵
20  END ↵
```

首先，敲 中文 键进入中文方式后系统自动处于“字母”状态，我们可以输入行号10和PRINT"。下面要输入汉字，因而敲 F_2 键进入拼音状态，打入“中”的拼音ZH，屏幕显示：

拼音：ZH　1这　2主　3中　4种　5着　6争

一看便知道需要的“中”字出现了。这时只要敲汉字前面的序号3，就把“中”字取到内存并显示在屏幕的相应位置上。凡是需要输入汉字都必须敲 F_2 ，使其转到拼音输入法的开始状态。而凡是输入其它符号都必须敲 F_1 键使其回到字符状态。上面的叙述过程可以用下面的敲键动作来表示：

] 中文

] [!/1] [)/0] [ ] [P] [R] [I] [N] [T]

[ ] [Shift]-["/,] [F_2] [Z] [H] [#/3] [F_2]

[H] [U] [A] [#/3] [F_2] [X] [!/1] [F_2] [X] [I]

[>/.] [>/.] [>/.] [>/.] [#/3] [F_2] [J] [∧/6] [F_1]

[Shift]—["/,] [RETURN]

] [@/2] [)/0] [ ] [E] [N] [D] [RETURN]

汉字的查找输入过程显示如下：

状态键	状态	拼音	供选择的汉字	敲键取字
F_2	拼音：	ZH	1这 2主 3中 4种 5着 6争	[#/3] 中
F_2	拼音：	HUA	1花 2哗 3华 4猾 5滑 6画	[#/3] 华
F_2	拼音：	X	1学 2下 3小 4线 5现 6性	[!/1] 学
F_2	拼音：	XI>.>.>.>.	1袭 2席 3习 4媳 5喜 6铣	[#/3] 习

[F_2] 拼音：J　　　1就 2级 3阶 4进 5加 6机 [∧ 6]机

敲[> .]键表示在同音字中继续查找后继汉字，敲[< ,]键就倒回到先前显示的汉字。

本节所述内容经常用到，建议读者面对一台中华学习机，亲自上机练习。

问题与思考

1. 一个 BASIC 程序中最多允许构造多少个不同的变量名？

2. 为什么变量名必须用字母开头？

3. 为什么算术表达式中的乘号既不能省略也不能简写？

4. 为什么要注意成对使用园括号？

习题

1. 分别写出三个不同的数值常数和字符串常数。

2. 下列字符序列哪些是正确的BASIC变量名？

XYZ，S1，END，S-1，SQRT，SUM，DATA，DTE

3. 写出下列数据的输出格式。

−328.75，+4235，$73.48_{10}{}^5$　−1234567892222

4. 将下列数学式写成BASIC表达式。

① $\frac{n(n-1)^2}{2}$　　② $\sqrt{x^{a+b^c}-\frac{a}{b\cdot c}}$

③ $\sqrt[3]{Ln\sqrt{1+d^2}-e^{2\beta}}$ ④ $2R\,Sin\frac{\beta}{2}$

⑤ $\dfrac{\cos x+\left|\dfrac{a-b}{a+c}\right|\cdot x^5}{e^{a^b}+e^{a^{b^x}}}$ ⑥ $\dfrac{\dfrac{a+1}{a+b}}{a-\dfrac{a+1}{a-\dfrac{a+1}{b}}}$

5. 用逻辑表达式刻划下列问题

①身高1.60且视力1.5。

②即是党员又是劳动模范。

③英语、数、理、化均在90分以上。

④数、理、化中有一门不及格。

⑤数、理、化中没有一门不及格。

6. 在机器上练习下列程序的输入。

```
10  INPUT A, B, C$↙
20  D=A+B↙
30  C$="我正在使用计算机"↙
40  PRINT C$, D↙
```

请写出输入程序时的敲键动作。

第三章　简单的BASIC程序

前已述及，BASIC程序是由若干语句组成的。那么BASIC语言中有些什么语句呢？它们的书写格式、作用如何呢？如何用语句来编写程序呢？从本章起，我们将由浅入深的来介绍BASIC的一些基本语句以及如何用这些语句来编写完整的能够在CEC-Ⅰ机上运行的程序。

3.1　赋值语句

在BASIC程序中，用得最多的语句可能就是赋值语句了。

3.1.1　赋值语句的格式和作用

格式：[LET]〈变量名〉=〈表达式〉

作用：将表达式的值送到变量中保存起来。

其中：LET是赋值语句的标志，实际使用中可以省略不写。变量名又称左部变量，它用于保存表达式的值。变量的类型可以是整型，或实型或串型。等号"="表示一种传递动作。表达式可以是任何合法的算术表达式或关系表达式或逻辑表达式，甚至也可以是字符串表达式。当然，等号两端的数据类型应当一致。

例3.1 求 $\dfrac{3\sqrt{100}}{\text{Ln}3+e^2}$ 的值

为便于读者练习，我们写一个完整的程序（行号之前的"]"是系统提示符，由机器自动显示，程序中不写）：

```
]10  W=3*SQR(100)/(LOG(3)+EXP(2)) ↵
]20  PRINT "W=";W ↵
]RUN ↵      (以下是运行结果)
W=3.5345396
```

这个程序包含了两个语句，语句行号分别为10和20。其中第10句就是一个赋值语句，它表示把3*SQR(100)/(LOG(3)+EXP(2))算术表达式的值计算出来再送到实型变量W中保存起来，以备后用。第20句是打印语句，表示把W的值显示在屏幕或打印在宽行纸上（PRINT语句后面还要专门介绍）。

RUN表示运行这段程序。运行程序之前，必须按要求将程序输入计算机内。对于这种简短的程序，建议读者直接在CEC-I机上输入，而不必用笔写在纸上后再输入。应当养成直接在计算机终端（键盘）上编写程序的习惯。

W=3.5345396是该程序运行后输出的结果。这里虽然只有两个语句，却是一个完整的能够在机器上运行的程序。

上例程序中赋值语句也可以写成：

```
10 LET  W=3*SQR(100)/(LOG(3)+EXP
(2)) ↵
```

例3.2
```
10  A=5 ↵
20  B=6 ↵
30  C=7 ↵
40  D=B*B-4*A*C ↵
50  PRINT  D ↵
RUN ↵   (以下是运行结果)
-104
```

前三句将5，6，7分别送到A，B，C变量中，第

40句利用前面获得的A，B，C的值来计算B^2-4AC的值（－104）并将其(－104)送到变量D中。第50句打印出D的值。

例3.1和例3.2　中赋值语句的左部变量和右端表达式都是数值型量。下面我们来看如何保存字符串。

例3.3　将字符串“大家都来学习和使用中华机。”传送到字符串变量W中。

```
10  W $=“大家都来学习和使用中华机。” ↲
20  PRINT  W $ ↲
RUN ↲       （以下是运行程序的结果）
大家都来学习和使用中华机。
```

例3.4　运行下列程序

```
10  S$ = “This is a book。” ↲
20  PRINT $ ↲
RUN ↲       （以下是运行结果）
This is a book.
```

例3.5　将字符串连接后打印出来。

```
10  H1 $=“张华” ↲
20  H2 $=“是” ↲
30  H3 $=“一个好学生。” ↲
40  H $=H1 $+H2 $+H3 $ ↲
50  PRINT H $ ↲
RUN ↲       （以下是运行结果）
张华是一个好学生。
```

10、20、30三个赋值语句的作用是把相应的汉字送到各自的字符串变量中。第40句是将三个字符串变量(H_1 $、$H_2$ $和$H_3$ $）的值进行连接运算，运算的结果仍然是一个字符

串“张华是一个好学生”，再将该字符串送到变量H$中。第50句打印出串变量H$的值（张华是一个好学生）。从这个例子也可以看出字符串运算符“+”和算术运算符“+”的本质区别。

从前面几个例子来看，行号均是从10开始的。提醒读者不要造成这样一个错觉：似乎BASIC程序的行号只能从10起编号。只要行号在63999以内，用任何整数开始作行号都可以。

语句行号的另一个规律读者可能早已发现，这就是行号始终是由小到大编号。这是正确的，也是必须的。BASIC语言中先执行的语句必须具有较小的行号。

对于程序的输入，我们可以按照语句行号由小到大的顺序依次输入机内，也可以随心所欲的按自认为合适的顺序输入机内。只要输入正确，对计算机来说效果完全一样。对于例3.5的程序，我们也可以按如下的语句顺序输入：

例3.6
```
40  H$=H1$+H2$+H3$ ↵
10  H1$= "张华" ↵
50  PRINT  H$ ↵
30  H3$= "一个好学生。" ↵
20  H2$= "是" ↵
RUN ↵ （以下是运行结果）
张华是一个好学生。
```

运行结果完全一样。为什么两个相同程序排列顺序完全不同以后运行的结果仍然相同呢？有两个原因：其一是语句行号没有变，只是输入语句的先后次序变了；其二是计算机总是从行号小的语句按行号由小到大顺序依次执行，尽管输入次序打乱了，但计算机执行语句的顺序仍然没有变。所以

运行结果完全相同。如果我们把行号和语句的对应关系打乱了，那运行结果一般说来是不同的。

例3.6 的作法尽管是正确的，却实在是不可取的，何必自己和自己过不去。

3.1.2 赋值语句的执行步骤

赋值语句写起来是简单的，但计算机系统在实现该语句功能时却必须采取三个步骤才能完成。

1． 确定左部变量

程序中给出一个赋值语句后，系统首先就要确定其左部变量。以上赋值语句中给出的左部变量都是简单变量的情况，因而确定左部变量并不麻烦。如果左部变量是下标变量（第六章介绍数组时将有详细说明），那么就必须先计算出下标后才能确定左部变量。在计算机系统内部，确定下标变量这种左部变量时还有一番复杂的计算，这里从略。有兴趣的读者请参阅编译技术方面的书籍。

2． 计算右端表达式的值

如果右端表达式是一个算术表达式，则按算术表达式的规则进行算术运算，最终得出一个数值；如果右端是字符串表达式，那么就按照字符串表达式的运算规则计算其值；右端也可以是逻辑表达式，相应计算的值就是逻辑值。下面是逻辑表达式作为右端项的例子：

例3.7 右端项是逻辑表达式的情形。

```
10  A＝8 ↲
20  B＝10 ↲
30  C＝7 ↲
40  L＝A＞4 ↲
50  T＝A*A＋B*B＞C∧4  AND L OR NOT B*
```

```
        B<100 ↵
60  PRINT L, T ↵
RUN ↵       （以下是运结行果）
1        1          （L和T均是逻辑值真）
```

在例3.7的程序中，40和50两句的右端都是逻辑表达达式。左部变量L和T均是逻辑变量。这个例子告诉我们，可以把逻辑表达式的值送到一个变量中保存起来，以后需用时就直接引用该逻辑变量。

3. 将表达式的值传送到左部变量中保存。

计算机要采取的第三个步骤就是将第二个步骤中计算出来的值送到由第一步所确定的变量中保存起来，后面的语句就可以直接引用变量名来使用其值了。

以上三个步骤，完全由计算机的硬件和软件来完成，不需要我们人工干予。在实际使用中，我们只需按要求写出正确的赋值语句就行了。事实上，机器在执行这三个步骤时，从直观上我们是感觉不到的。

3.1.3 使用赋值语句时的注意事项

赋值语句在实际程序设计中用得最多，几乎任何一个有意义的BASIC程序都不可能没有赋值语句。但使用时必须注意以下几点：

1. 左部变量和右端表达式的数据类型必须一致。

如果右端是算术表达式，那么左部变量只能是数值型变量。如果右端是字符串表达式，那么左部变量只能是串型变量（标志是$）。如果右端是逻辑表达式，那么左部变量只能是逻辑变量。逻辑变量同实型变量一样没有任何外观标志，但系统内部是清楚的。

如果赋值语句等号两端的数据类型（主要是指算术类、

逻辑类和字符串类）不一致，那么执行该语句时就会发生数据类型不匹配的错误。

例3.8 执行下面的赋值语句。

```
10  D$=3+LOG(5) ↵
20  PRINT D$ ↵
RUN↵              （以下是运行结果）
? Type mismatch error in 10
```

运行这个程序后，在第10句发生了类型不匹配的错误（Type mismatch erorr in 10）。我们重新检查程序就会发现，在第10句的赋值语句中，左部变量是串型变量，而右端却是一个算术表达式，理所当然的要发生数据类型错误。改正的办法是根据实际情况将左部变量换成实型量或将右端写成字符串形式。如果左部和右部只是实型和整型的区别，那么机器将自动进行这种转换。

如果赋值语句的两端均是算术量（仅仅是整实之间的差别），那么机器将自动进行这种转换。转换的原则是：如果左部是整型量（标志是%）而右部是实型量，那么机器自动将右部实型量取整后按整数形式保存到左部变量中。（当一个实型数转换成整型数时，取小于或等于该实数的最大整数作为转换后的结果）。如果左部是实型量而右端是整型量，那么机器自动将整型值转换成实型值后保存在左部变量中。

3. 左部变量只能是一个不带运算符的变量名，而不能是常数或带运算符的表达式。

例3.9 判断下列赋值语句是否正确？

```
10  5=3*x+y ↵
20  -A=8 ↵
30  x+y=SQR(100)+LOG(5) ↵
```

```
40  PRINT A, X, Y ↲
RUN ↲
? Sytax error          (语法错误)
```

显然，上述程序中10、20和30句中的赋值语句均是错误的。左部变量不正确。改正的办法是将它们各自的左部都换成变量名。

4. 赋值语句的标志LET可以省去不写。

前面所举的赋值语句的例子中已经这样作了。LET省去后不会对语句有影响。

5. 等号"＝"的作用。

赋值语句中的等号"＝"，既不同于数学中的"＝"，又不同于关系表达式中的"＝"。数学中的等号代表的是一个相等关系，左右两端可交换。而关系表达式中的"＝"代表的是一种相等比较运算，运算的结果是一个逻辑值。赋值语句中的"＝"，代表的是一种传送的操作动作。请看下面的例子：

例3.10
```
10  X＝3 ↲
20  X＝X＋5 ↲
30  L＝X＝5 ↲
40  PRINT X, L ↲
RUN ↲               (以下是运行结果)
 8              0
```

对于第20句的X＝X＋5，在数学中看来这是完全不可能的，因为不会有任何一个变量加上一个数值后仍然等于变量本身。但BASIC语言是允许的。它表示在原来基础上累加一个值后再送到原单元中保存。

第30句是一个逻辑赋值语句，它表示把X＝5的逻辑值

送到逻辑变量L中保存。这一句中有两个等号，前一个“=”代表传送操作，后一个“=”代表关系运算。

从输出结果来看，8是X累加上5后得到的值。而0是逻辑值假，因为X的值并不等于5，所以L的值是假(0)。

6．可以利用赋值语句进行累加和连乘。

例3.11 运行下列程序。

```
10  A=3 ↲
20  W=0 ↲
30  S=1 ↲
40  W=W+A ↲
50  W=W+A ↲
60  S=S*A ↲
70  S=S*A ↲
80  PRINT A, W, S ↲
RUN ↲                （以下是运行结果）
3          6          9
```

40～50句就将A的值累加起来了，因为累加了两次，所以输出结果为6。60～70句把A的值连乘了两次，所以输出结果为9。

7．赋值语句也是使变量获得数据的一种方式。

例3.12 运行下列程序。

```
10  A=5 ↲
20  B=6 ↲
30  C=7 ↲
40  PRINT A, B, C ↲
RUN ↲
5          6          7
```

显然A、B、C的值5、6、7就是从赋值语句得来的。如果数据个数较少，或许用赋值语句来使变量获得值还是一种好方法。

3.2 键盘输入语句

赋值语句可以使变量获得数据这是毫无疑问的。但是，如果把赋值语句作为变量获得数据的唯一方式，那么读者将会发现，输入100个不同的数据就得有100个不同的赋值语句。那泛味的程序不就太长了吗？是的，对实用来说可能是不方便的。因而BASIC语言提供了多种输入语句。

3.2.1 键盘输入语句的格式及其作用

格式：INPUT [“〈提示〉”;]〈变量名表〉

作用：允许用户从键盘上接收数据并送到对应变量中。

其中：待选项[〈提示〉;]是用户自己给出的提醒输入数据的标记，它由字符串组成。变量表是希望从键盘上接收数据的一个或多个变量名。如果表中有多个变量名，那么彼此之间要用逗号“，”隔开。

例3.13 用键盘语句输入A，B，C的值。

```
10  INPUT“请输入数据：”；A，B，C ↵
20  D=B*B-4*A*C ↵
30  PRINT A，B，C ↵
40  PRINT D ↵
RUN ↵
请输入数据：5，6，7 ↵
5           6           7
-104            (运行结果)
```

对于这个程序需要作如下的说明：

①请输入数据：这是为提醒用户注意而给出的提示信息。在实际应用中，用户可以给出自己所需要的提示信息。

② 提示信息之后一定要有一个分号“；”，其作用是将提示信息和变量表隔开。

③ INPUT之后的A，B，C是变量表，表中有A、B、C三个变量名。A、B和B、C间都用逗号隔开。

④ 打入RUN↲命令后，屏幕上显示：请输入数据：这几个字符，并等待用户输入数据。本例中输入是5，6，7三个常数。至此，机器自动把5送到A中，6送到B中，7送到C中。输入的数据间要用逗号“，”隔开（请读者考虑为什么要用“，”隔开）。最后的5，6，7和－104是输出的运行结果（30，40两句的作用）。

例3.14 键盘输入数据后运行程序。

求 $V=\frac{\pi h}{6}\sqrt{3(a^2+b^2+h^2)}$ 的值，其中

$\pi=3.14159$，$a=5.3$，$b=2.7$，$h=3.72$

```
10  INPUT "输入数据："; BA, A, B, H ↲
20  V=BA*H*SQR( 3 *(A*A+B*B+H*H))
        /6 ↲
30  PRINT BA, A, B↲
40  PRINT H, V ↲
RUN ↲
输入数据：3.14159, 5.3, 2.7, 3.72 ↲
3.14159             5.3             2.7
3.72                23.6942679
```

上述程序也可以用赋值语句使变量获得数据，为便于读

者比较，请看下例。

例3.15 用赋值语句使变量得到数据。

```
10  BA=3.14159 ↵
20  A=5.3 ↵
30  B=2.7 ↵
40  H=3.72 ↵
50  V=BA*H*SQR(3*(A*A+B*B+H*H))
    /6 ↵
60  PRINT BA, A, B ↵
70  PRINT H, V ↵
RUN ↵
3.14159          5.3              2.7
3.72             23.6942679
```

运行结果完全一样。为使程序简短，也可以把10—40句合成一句：

```
10  BA=3.14159:A=5.3:B=2.7:H=3.72 ↵
```

以上例子说明，对同一个问题可以用多种语句来解决，但必须保证所得结果完全相同。

在INPUT语句中，如果省略了提示信息，那么系统用一个问号“?”作为提示用户输入数据的标志。参看下例。

例3.16 运行下列程序并输入数据。

```
10  INPUT BA, A, B, H ↵
20  V=BA*H*SQR(3*(A*A+B*B+H*H))
    /6 ↵
30  PRINT BA, A, B ↵
40  PRINT H, V ↵
RUN ↵
```

? 3.14159，5.3，2.7，3.72 ↵ （输入数据）

3.14159　　　　　5.3　　　　　2.7

3.72　　　　　　23.6942679

其中的“？”就是系统给出的提醒输入数据的信息。

3.2.2 使用INPUT时的注意事项

1．变量名必须和输入的数据一一对应。

如果变量名和所敲入的数据不对应，那么系统是不容易发现这类错误的。因此，我们使用时应倍加注意，否则将导致错误的计算结果。如果有多个变量名，变量名之间必须用逗号隔开。如前面示例中的INPUT A，B，C所示。

2．变量名和输入数据的类型必须一致。

如果变量名和输入数据类型不正确，则屏幕显示“? REENTER”提示，意思是要求用户重新敲入正确的数据。

3．用INPUT接收字符串时，要求从键盘上敲入的字符串中不能含有逗号。因为逗号是用于分隔数据的标志。

4．用INPUT接收的只能是具体的常数（可以是数值常数或字符串常数），而不能是变量名或表达式。参看下例。

例3.17 键盘敲入数据。

```
10 W=5 ↵
20 INPUT "输入数据：";A，B，C ↵
30 D=(A+B)/C ↵
40 PRINT D ↵
RUN ↵
输入数据：W，SQR(100)，5+3 ↵
```

显然敲入的数据不正确，因为W，SQR(100)，5+3均不是常数而是表达式。

5. 如果INPUT中接收的是多个数据，那么在敲入数据时，可以用逗号来分隔数据，也可以用↲来分隔数据。如果用“，”分隔，最后数据之后必须敲↲键，在此情况下，↲作为所有数据是否完结的标志。

例3.18 用↲来分隔敲入的多个数据。

```
10  INPUT A, B, C ↲
20  D=B*B-4*A*C ↲
30  PRINT A, B, C ↲
40  PRINT D ↲
RUN ↲
? 5 ↲              (用回车键↲来分隔数据)
?? 6 ↲
?? 7 ↲
 5            6            7
-104           (运行结果)
```

6. 如果INPUT中给出了提示信息，那么提示信息之后一定要有一个分号“；”，它用于把提示信息和变量表分隔开。

7. 一个算术变量在没有获得值之前其值为0，一个字符串变量在未得到值之前为空。

3.2.3 GET语句

格式：GET〈变量名〉

作用：允许用户从键盘上接收一个字符到指定的变量中。这里的字符根据变量名的类型来确定，可以是数字，也可以是其它字符。

例3.19 用GET将数字3送到A中。

```
10  GET A ↲
```

```
20  PRINT A ↵
RUN ↵
 3            (敲入数据字符3)
 3            (打印的结果)
```

使用GET来接收数据时注意：

1． 机器执行GET语句时屏幕上没有任何提示标记，输入的字符也不显示在屏幕上。

2． 敲入数据之后不必敲↵键，因为只接收一个字符，不用敲↵标志机器也能判断数据是否结束。

3． 如果GET之后的变量名是算术变量，那么敲入＋－·，：符号后变量的值均为零。而输入其它字符时均发生语法错，这时屏幕显示“？Syntax error”且程序停止执行。

4． 用GET来接收数据时，无论在键盘上敲入了多少个字符，变量均只能接收第一个字符。

通常，实际使用中都不大喜欢用GET来接收参与计算的数据（因为参与计算的数据往往不止一位数字），而大多用于接收一个固定的字符，以便用于控制程序的执行流程。

3.3 输出语句

前面的例子中我们已经用到了输出语句PRINT，但没有作详细介绍，现在是该详细介绍的时候了。

3.3.1 输出语句的格式和作用

格式：PRINT〈表达式表〉　或？〈表达式表〉

作用：将表达式的值显示在屏幕或打印在宽行纸上。程序中也可以用？代替PRINT，效果完全相同。

这里的表达式表是需要打印其值的一个或多个表达式。表达式表中的各个表达式之间用逗号“,”隔开，也可以用分号“;”隔开。但用“,”和“;”隔开时，打印出来的数据的排列方式略有不同。

例3.20　PRINT语句的用法。

```
10  INPUT “输入数据:”; A, B, C ↲
20  D=B*B-4*A*C ↲
30  PRINT A, B, C ↲
40  PRINT D ↲
RUN ↲
输入数据: 35.42, 72.83, -555.444 ↲
35.42              72.83              -555.444
83999.5148
```

第30句的输出语句中用“,”作了A，B，C彼此间的分隔符。第40句因为只有一个D，因而不用分隔符。

在PRINT中的表达式也可以是字符串表达式。如下例。

例3.21　用X=xx形式打印出表达式的值。

```
10  INPUT “输入数据:”; X, Y, Z ↲
20  W=SQR(X*X+Y*Y+Z*Z) ↲
30  PRINT “X=”;X, “Y=”;Y, “Z=”;Z ↲
40  PRINT “W=”;W ↲
RUN ↲
输入数据: 5, 6, 7 ↲
X= 5               Y= 6               Z= 7
W=10.4880885
```

第30句的PRINT中，既有字符串表达式又有算术表达

式。

可以用PRINT直接输出表达式的值。

例3.22 运行下列程序。

```
10 INPUT "输入数据:"; X, Y, Z ↲
20 PRINT "W="; SQR(X*X+Y*Y+Z*Z)↲
RUN ↲
输入数据: 5, 6, 7 ↲
W=10.4880885
```

第20句中PRINT作用是先计算表达式SQR(X*X+Y*Y+Z*Z)的值，再将其值打印出来。如果一个表达式的值不在其它表达式中引用，那么用PRINT直接打印出来是可以的。否则，最好用赋值语句保存值后再打印。

3.3.2 分隔符"，"和"；"在PRINT中的用法

1. 分隔符"，"

在PRINT语句中，如果输出表达式中用了一个逗号，那么在输出时，逗号之后的表达式的值将占用16个字符位置。

例3.23 求$\left(\frac{a}{b}+\frac{c\cdot d}{e}\right)^3$的值，其中a=-123.78，b=1324，c=54.32，d=123.3，e=7.82

```
10 INPUT "输入数据:"; A,B,C,D,E ↲
20 F=(A/B+C*D/E)∧3 ↲
30 PRINT "A", "B", "C" ↲
40 PRINT A,B,C ↲
50 PRINT "D", "E", "F" ↲
60 PRINT D,E,F ↲
RUN ↲
```

输入数据：−123.78，1324，54.32，123.3，7.82 ↵

A	B	C
−123.78	1324	54.32
D	E	F
123.3	7.82	62806708

上例输出数据的排列格式是这样的：

字母A占据1～16列，字母B占据17～32列，字母C占据33～40列。同样，−123.78占据新行上的1～16列，1324占据～1732列，54.32占据33～40列。其余类似。

在PRINT中，如果我们用逗号作为分隔符，那么系统按16位分为一组，将输出宽度为40个字符的屏幕分成三组，前两组各占16位，第三组占8位。如果输出的第一个表达式的值占满了16位，那么下一个表达式的输出值就占据第三组的8位。如果第三个输出值需要占据7位以上的字符位置，那么系统将把第三个输出值自动推到下一行的第一个字符位置开始输出。这种格式称为标准输出格式。我们在准备输出数据时，应当注意数据位数不要超过15位。在第三组区输出时不要超过7个字符。

2. 分隔符“；”

在PRINT中，如果用“；”作分隔符，那么数据按紧凑格式输出。

例3.24 按紧凑格式输出表达式的值。

```
10  INPUT“输入数据:”;A,B,C,D,E ↵
20  F=(A/B+C*D/E)∧3 ↵
30  PRINT A;B;C;D;E;F ↵
RUN ↵
```

输入数据：−123.78，1324，54.32，123.3，7.82 ↵

−123.87 132454.32 123.3 7.82 62806708

从上面结果可以看到：紧凑格式输出的数据之间没有空格，以致造成不大好辩认各个数据。如果在每两个数据之间输出一个空格，那么可读性就强多了。请读者自己完成。

3.3.3 打印空行和换行打印

1. 打印空行

如果输出数据每两行之间希望留有较宽的空白，那么可以在输出数据时有意打印一个空行。

例3.25 打印空行

```
10  INPUT "输入数据:";A,B,C,D,E ↵
20  F=(A/B+C*D/E/)∧3 ↵
30  PRINT A,B,C ↵
35  PRINT ↵
40  PRINT D, E, F ↵
RUN ↵
输入数据：−123.78, 1324, 54.32, 123.3, 7.82 ↵
−123.78              1324              54.32
123.3                7.82              62806708
```

上述程序中第35句，打印的就是一个空行，其作用相当于把打印机的打印头向下移动了一行。因而在输出数据间留了一个空白行。也可以把35句改为：35　PRINT“ ”，效果完全一样。

2. 换行打印

如果需要把数据打印在不同的行上，可以有两种换行的办法。一是人为的安排换行打印；二是由机器自动换行。

我们先来看下面的例子。

例3.27 按两个数据一行打印数据。

```
10  INPUT "输入数据:";A, B, C, D, E ↵
20  F=(A/B+C*D/E)∧3 ↵
30  PRINT A, B ↵
40  PRINT C, D ↵
50  PRINT E, F ↵
RUN ↵
输入数据:-123.78, 1324, 54.32, 123.3, 7.82 ↵
-123.78              1324
54.32                123.3
7.82                 62806708
```

第30句表示打印出A，B的值以后换行，即下一个打印语句打印在一个新的行上。40，50两句作用类似。归纳起来，如果一个PRINT语句的最后那个表达式（如上例的B，D，F）之后既没有逗号也没有分号（直接跟↵），就表示输出其值后换行。如果跟有逗号或分号就不换行。参见下例。

例3.27　不换行打印。

```
10  INPUT A, B, C, D, E ↵
20  F=(A/B+C*D/E)∧3 ↵
30  PRINT A;"␣␣";B;"␣␣";C; ↵
40  PRINT "␣␣";D;"␣␣;E;"␣␣";F ↵
RUN ↵
? -123.78, 1324, 54.32, 123.3, 7.82 ↵
  -123.78  1324  54.32  123.3  7.82  62806708
```

在第30句中，由于C之后有一个分号，就表示打印完C的值后不换行，第40句打印的值紧跟其后。尽管30和40分别是两行上的两个打印语句，但它们打印的数据却安排在同一行上。因此，打印语句的个数和所占行数不能确定打印数据占

用的行数。示例中的一个“␣”表示一个空格。有时为了准确描述打印格式，用“␣”明确表示出空格是个好办法。当然C之后也可以跟逗号，同样表示不换行，但打印格式就必须按逗号的要求办理。

实际上我们也可以不必人为换行，而由机器自动调整换行。

例3.28 机器自动换行打印。

```
10 INPUT A, B, C, D, E ↙
20 F=(A/B+C*D/E)∧3 ↙
30 PRINT A, B, C, D, E, F ↙
RUN ↙
? -123.78, 1324, 54.32, 123.3, 7.82 ↙
 -123.78          1324          54.32
  123.3           7.82          62806708
```

当所有表达式的值在一行上打印不下时，机器自动转到下一个新行上继续打印。本例中第30句就是如此。

3.4 程序实例

我们已经介绍了赋值语句、键盘输入语句和打印语句，利用这些语句就可以编写一个简单而完整的程序了。为加深读者对程序设计的理解，我们再举一个例子。

例3.29 求$\frac{e^{ax}\cos(2bx+c)}{2a}-$

$$\frac{e^{ax}[a\cos(2bx+c)+2b\sin(2bx+c)]}{2(a^2+4b^2)}$$ 的值。

其中：a=3，b=4，c=5，x=6

```
10  INPUT "输入数据:";A, B, C, X ↵
20  W1=EXP(A*X)*COS(2*B*X+C)/
    (2*A) ↵
30  W2=EXP(A*X)*(A*COS(2*B*X+C)
    +2*B*SIN(2*B*X+C))/(2*(A*A+
    4*B*B)) ↵
40  W=W1-W2 ↵
50  PRINT A, B, C ↵
60  PRINT X, W ↵
RUN ↵
输入数据: 3, 4, 5, 6 ↵
3               4               5
6               -10234605.6
```

这个程序虽然正确，但从程序设计的技术来说手法并不高明。最明显的缺陷就是2 * B*X+C式子重复了三次而没有用一个变量来代替。这不但增加了程序的书写麻烦，也浪费了机器的执行时间。一个稍加改进的程序如下：

例3.30 例3.29程序的一个改进程序。

```
10  INPUT A, B, C, X ↵
20  W1=EXP(A*X) ↵
30  W2=2*B*X+C ↵
40  W3=COS(W2) ↵
50  W4=SIN(W2) ↵
60  W=W1*W3/(2*A)-(W1*(A*W3+2*
    B*W4))/(2*(A*A+4*B*B)) ↵
70  PRINT A, B, C ↵
80  PRINT X, "W=";W ↵
```

```
RUN ↵
? 3, 4, 5, 6 ↵
3             4                        5
6             W=-10234605.6
```

当然,这个所谓改进后的程序仍然不够令人满意,最大的问题就是直观性差了,因为程序中引入了一些中间变量。对初学者来说,我们首先要求编写程序正确,只要熟练自然就能生巧。从上面给出的示例中,我们已经体会到编制程序的一些初步方法:

1. 熟悉问题,弄懂解题方法。如果问题的解决方法都不清楚,那无论如何是编不出程序来的。

2. 给变量以正确的命名。实际问题中给出的名字并不一定符合BASIC语言的要求。因而要对实际问题中的变量进行改造,如希腊字母或罗马文字母改用相近的英文或汉语拼音字符,使之合法化。

3. 重复出现的式子用中间工作变量代替,以备后用。

4. 变量用什么方式获得数据,这里仅介绍了INPUT、GET和赋值语句的办法,以后还会介绍另外的办法。总的原则是:简单和方便。

5. 数据的输出格式。总的原则是:数据格式要便于阅读和浏览,尽量照顾人们通常的习惯。

3.5 常用命令和误操作的修改

3.5.1 常用命令

这里所谓的常用命令是操作系统提供给BASIC系统使用的一些命令。有些命令可以当成BASIC的语句来使用,而

有些命令则只能作为键盘操作命令来使用。

1、程序结束标志END

格式：END

作用：使程序终止执行，系统不提供任何提示信息。

用法：使命令可以象BASIC语句一样写在程序需要终止的地方，表示程序执行到此完结。

例3.31 带有END标志的程序。

```
10  INPUT A, B, C ↲
20  F=(A*A+B*B)/C ↲
30  PRINT "F=";F ↲
40  END ↲
RUN ↲
? 4, 3, 5 ↲
F= 5
```

如果读者高兴的话，本章的所有示例程序的末尾均可以加上程序结束标志END。

2. 运行程序命令RUN

格式：RUN 或RUN〈行号〉

作用：运行内存中的BASIC程序。如果带有行号，则从指定行号开始运行。

用法：一个BASIC程序输入到内存以内，如果我们希望立即执行之，则在程序末尾敲入RUN命令，机器就立即执行输入到内存中的BASIC程序。示例请参见本章例子。

3. 列出程序命令LIST

格式：LIST 或LIST [〈行号—1〉] [,〈行号—2〉]

作用：在屏幕上显示内存的BASIC程序。

用法：我们输入一个BASIC程序后，如果希望查看程序输入是否正确，那么就可以用该命令列出程序。如果不指出行号，则列出全部程序清单。如果指出了行号，那么就列出从行号-1到行号-2之间的程序段。如] LIST ↵就可显示出内存中保存的BASIC程序。

4. 暂停屏幕画面翻动命令CTRL—S

格式：CTRL—S键

作用：暂停屏幕画面卷动。

用法：CTRL—S实际上是键盘上的控制键。如果一个程序很长，屏幕一次显示不下，那么用LIST列程序时，程序就不断地向上翻动，最终显示在屏幕上的只是程序末尾的那么十几行。为了使屏幕的翻卷停止，可以敲CTRL—S键。待程序看清楚后再敲CTRL—S，又继续翻卷。

5. 删除程序的命令NEW

格式：NEW

作用：删去内存中的BASIC程序。

用法：如果我们要输入一个新的程序，就应当把原来不用的程序删去。否则，新老程序就混在一起，影响运算的正确性。清除旧程序可以直接敲入NEW命令。如果你要检查旧程序是否已被删去，可用LIST命令。

以上是经常用到的一些命令和操作，其它还有一些命令请参见第十二章操作系统的有关节段。

3.5.2 误操作的修改

所谓误操作是指输入程序时敲错了键，因而导致程序运行出错。即使是程序输入的老手误操作有时也在所难免，初学者就更是难以避免了。现在介绍两种更正输入出错的方法。

在输入某行程序时，如果在敲↵键之前就发现某个字符

敲错了，那么可以敲键盘上的◁键，将光标（又称游标）移到出错字符处，然后重新输入出错字符之后的所有字符。

如果换行以后发现某行的某个字符输入有误，干脆就在一个新行上将有误的行重新输入，这对初学者来说是最简单的修改办法。

还有其它一些修改方法，如编辑状态下的修改。这里从略。

问题与思考

1． 哪些情况下用赋值语句使变量获得数据比用INPUT更方便？

2． INPUT和GET语句的主要区别是什么？

3． 赋值语句和表达式有什么关系？

4． 逗号在哪些地方用到？其作用如何？

5． 一个语句放在程序的任何位置上其运行结果均相同吗？为什么？举出一个例子。

习题

1． 写出下列程序的运行结果。

①
```
10  X=10 ↵
20  Y=20 ↵
30  Z=30 ↵
40  X=Y/2 ↵
50  Y=Z/3 ↵
60  X=X+Y+Z ↵
70  Z=Z/3 ↵
80  X=X/2 ↵
90  Y=X*2 ↵
100 Z=X+Y+Z ↵
```

```
110 PRINT X, Y, Z ↵
120  END ↵
RUN ↵
```

②

```
10  INPUT X, Y, Z ↵
20  X=X+Y+Z ↵
30  Y=X+Y+Z ↵
40  Z=X+Y+Z ↵
50  PRINT X, Y, Z ↵
60  END ↵
?  3 , 6 , 9 ↵
```

③

```
10  PRINT X, Y, Z ↵
20  X=10 ↵
30  Y=20 ↵
40  Z=30 ↵
50  PRINT X+Y+Z ↵
60  END ↵
RUN ↵
```

④

```
10  GET X ↵
20  GET D$ ↵
30  W=X+5 ↵
40  PRINT D$; "=" ;W ↵
50  END ↵
RUN
50
WST
```

⑤

```
10  INPUT X, Y, Z ↵
20  W1=X+Y+Z ↵
```

```
30  W2%=X+Y+Z ↵
40  PRINT W1, W2% ↵
50  END ↵
RUN ↵
? 3.5, 4.8, 7.3 ↵
```

2．按格式写出下列程序运行的结果

```
10  INPUT"请输入数据："; A, B, C, D, E, F,
    G, H ↵
20  W=A+B+C+D+E+F+G+H ↵
30  PRINT A, B, C, D, E, F, G, H ↵
40  PRINT ↵
50  PRINT "A", "B", "C", "D" ↵
60  PRINT A, B, C, D ↵
70  PRINT "E", "F", "G", "H" ↵
80  PRINT E, F, G, H ↵
90  PRINT "A=";A, "B=";B, "C=";C ↵
100 PRINT D; E; F; G; H; ↵
110 PRINT "W=";W ↵
120 END ↵
RUN ↵
请输入数据：1.111111111, 2.222, 3.333, 4.444,
5.555, 6.66, 2, 5 ↵
```

3．指出下列程序的书写错误。

求$Me^{-\beta t}\cos(\omega t+\phi)$的值，其中M=10，β=3.5，t=3，ω=30°，φ=5，程序如下：

```
10  INPUT M, t ↵
20  ω=30° ↵
```

```
30  φ= 5 ↵
40  H=M * E∧(-β * T) * COS(ω * T+φ) ↵
50  β=3.5 ↵
60  PRINT H, β ↵
70  END ↵
RUN ↵
?  10 3.5 ↵
```

4. 编写下列问题的程序。

①求$\left(\frac{a-c}{a+b}+x\right)^2+\sqrt{y^2+4x^2\ \sin\beta x}\div e^4x^2$的值（自己给出所需数据）。

② 按下列格式打印表格（编写出程序）。

姓名	性别	年龄	班级
张扬	男	15	初二.一
李芳	女	14	初二.二

……

第四章　转向和分支程序

在实际应用中，有时候希望跳过一些语句或根据某种条件来选择执行某些语句　这就是要用到转向和条件分支。BASIC为达到这种目的，提供了转向语句和条件语句。

4.1　转向语句

格式：GOTO　〈行号〉

作用：无条件转到指定行号的语句去执行。其中：GOTO是转向语句的标志，行号是程序中需要转去的语句行号。转向语句使我们能够多次地执行某些语句，也可以使我们跳过某些语句。

我们来看下面的例子（为了简洁起见，从本章起，语句末尾的回车键↵省写，但在输入程序时一定要敲入，切记）。

例4.1　分别给出三组A、B、C的值，求

$$\left(A+B^{C}-\frac{A}{BC}\right)/\left(A^{2}+B^{2}\right)\text{的值。}$$

```
5 I=0
10 INPUT “输入数据：”；A，B，C
20 I=I+1
30 W=（A+B∧C−A/B/C）/（A*A+B*B）
40 PRINT “第”；I；“组的值”
50 PRINT “A=”；A；“ ”；“B=”；B；“ ”；
```

```
    "C=" ; C; " " ; "W=" ; W
60 GOTO 10
70 END
RUN↲              （以下是运行结果）
输入数据：3，4，5↲
第1组的值
A=3  B=4  C=5  W=41.074
输入数据：7，8，9↲
第2组的值
A=7  B=8  C=9  W=1187767.57
输入数据：1.35，3.24，4.78↲
第3组的值
A=1.35  B=3.24  C=4.78  W=22.4789377
CTRL-C            （中断执行的控制键）
Break in 10       （在10句中断）
]
```

在这个例子中，首先输入第一组值，经过计算后分别输出A、B、C、W的值。当执行到第60句时，又转回到第10句，要求输入第二组值。如此重复，直到三组值输入完毕。但由于60句是一个无条件转向语句，因而尽管三组值已输完，但程序仍然要求再输入数据。这时，如果希望程序停止执行，只好用CTRL-C键，使程序强行中断回到"]"提示状态。

程序比较简单。其中的变量I是用于计数的，用于统计输入了几组值。

4.2 条件语句

对于例4.1的程序，我们已指出它是不能自动停止执行的（除非我们不输入数据），原因就在于使用了一个无条件转向语句GOTO 10，导致它老是转向第10句。但现实使用中，总希望执行若干次以后能够自动停下来，因而需要用到条件语句。

4.2.1 条件语句的格式和作用

1． IF-GOTO型条件转向语句

格式：IF〈条件〉GOTO〈行号〉

作用：如果条件成立，就把控制转到指定的行号去执行。否则，执行该语句的后继语句。

所谓条件成立与否，是这个意思：

如果条件是关系表达式或逻辑表达式，只要表达式的值为真，那么就称条件成立。否则称为条件不成立（即关系表达式的值为假）。

如果条件是算术表达式，只要算术表达式的值不为零（若表达式的绝对值小于或等于2.9387E-39，则作为零处理），就称条件成立。否则称为不成立（即算术表达式的值等于零）。

如果条件是字符串表达式，只要字符串表达式的值不是空串，就表示条件成立。否则就表示条件不成立（即字符串表达式的值为空格）。

这里对条件成立与否的解释，适用于整个CEC-BASIC的条件。

我们用例子来说明条件语句的用法。

例4.2　为便于对照，仍用例4.1为例。

IF中条件是关系表达式的情况：

```
5  I=0
10 INPUT "输入数据："；A，B，C
20 I=I+1
30 W=(A+B∧C-A/B/C)/(A*A+B*B)
40 PRINT "第"；I；"组的值"
50 PRINT "A="；A；" "；"B="；B；" "；
 "C="；C；" "；"W="；W
60 IF I<3 GOTO 10
70 PRINT "程序结束"
80 END
RUN↙
输入数据：3，4，5↙
第1组的值
A=3   B=4   C=5   W=41.074
输入数据：7，8，9↙
第2组的值
A=7   B=8   C=9   W=1187767.57
输入数据：1.35，3.24，4.78↙
第3组的值
A=1.35  B=3.24  C=4.78  W=22.4789377
程序结束
]
```

在第60句，如果I＜3为真，就再转到第10句输入一组值并执行计算和输出。当I=3时（实际已输入了三组值），条件不成立，就自动执行第60句的后继语句70 PRINT "程

序结束”，打印出一个程序结束标志，提示用户，需要输入和计算的三组数据已经完成。这时系统返回到“]”提示状态。

例4.3 IF中条件为算术表达式。

```
5  I=0
10 INPUT "输入数据：";A, B, C
20 I=I+1
30 W=(A+B∧C-A/B/C)/(A*A+B*B)
40 PRINT "第";I;"组的值"
50 PRINT "A=";A;" ";"B=";B;" ";
 "C=";C;" ";"W=";W
55 INPUT "输入K值：";K
60 IF K GOTO 10
70 PRINT "程序结束"
80 END
RUN↵
输入数据：3，4，5↵
第1组的值
A=3   B=4   C=5   W=41.074
输入K值：9↵           （输入非零值均可）
输入数据：7，8，9↵
第2组的值
A=7   B=8   C=9   W=1187767.57
输入K值：5↵           （输入非零值）
输入数据：1.35，3.24，4.78↵
第3组的值
A=1.35  B=3.24  C=4.78  W=22.4789377
```

输入K值：0↵　　　（输入0值，结束运行）

程序结束

第55句增加了输入一个算术值K，以确定是否需要继续输入数据。当执行到第60句时，如果K的值不为零，那么就转到第10句继续输入下一组值。如果K的值为零，就执行后继语句70，打印程序结束标志。

这个程序的执行次数完全由用户控制，比起前面的例4.2来，似乎灵活些。

例4.4　IF中条件为字符串表达式。

```
5 I=0
10 INPUT "输入数据：";A，B，C
20 I=I+1
30 W=(A+B∧C-A/B/C)/(A*A+B*B)
40 PRINT "第"；I；"组的值"
50 PRINT "A=";A；" "；"B="；B；" "；
 "C="；C；" "；"W="；W
55 INPUT "输入K$值："；K$
60 IF K$ GOTO 10
70 PRINT "程序结束"
80 END
RUN↵
输入数据：3，4，5↵
第1组的值
A=3  B=4  C=5  W=41.074
输入K$值：M↵        （敲入任何非空格字符）
输入数据：7，8，9↵
第二组的值
```

A＝7　B＝8　C＝9　W＝1187767.57

输入K$值：0↵　　　（敲非空格字符，继续运行）

输入数据：1.35，3.24，4.78↵

第3组的值

A＝1.35　B＝3.24　C＝4.78　W＝22.4789377

输入K$值：空格↵　　（敲入空格字符，结束运行）

程序结束

这里的执行情况类似算术表达式，不再说明。

2． IF-THEN型条件语句

格式：IF〈条件〉THEN〈行号〉

作用：如果条件成立，则执行THEN之后指定的行号语句。否则，顺序执行该语句的后继语句。

这里的条件，其含义同前述，它可以是关系表达式、逻辑表达式、算术表达式或字符串表达式。行号必须是程序中出现的行号。这种形式的条件语句同IF〈条件〉GOTO〈行号〉形式的条件语句用法完全相同。在前面例4.2～4.4的例子中，只要把程序中的IF-GOTO换成IF-THEN均能正确运行。请读者自己完成。

3． IF-THEN〈语句〉型条件语句

格式：IF〈条件〉THEN〈语句序列〉

作用：如果条件成立，则执行THEN之后给定的语句序列，如果没有转向语句，执行完给定语句序列后顺序执行后继语句。如果条件不成立，就直接执行后继语句。

这里的语句序列可以是一个或多个语句。若是多个语句，语句彼此之间用冒号"："隔开。

例4.5　求W的值。

$$W = \begin{cases} \frac{x}{a} e^{-\frac{x^2}{2a^2}} \cdot \mathrm{Sin}x & x > 0 \text{ 且 } a \neq 0 \\ \cos(ax+5) & \text{其它} \end{cases}$$

编出程序如下：

```
10 INPUT "输入数据："；A, X
20 W=COS(A*X+5)
30 IF A<>0 AND X>0 THEN
   W=X/A*EXP(-X*X/(2*A*A))*
   SIN(X)
40 PRINT "A="；A；" "；"X="；X；" "；
                    "W="；W
50 PRINT "程序结束"
60 END
RUN↵
输入数据：15, 21↵
A=15   X=21   W=.439608606
程序结束
```

程序执行情况是这样的：

执行第10句首先输入A、X的值（本例中为15和21）。因为W的取值只有两种情况，因而先假设一种条件满足，这就是第20句的作用。第30句再来判断给定条件是否成立。若条件成立，则W又送入了X/A*EXP(-X*X/(2*A*A))*SIX(X)的值，这就把先前20句送入的假定值冲掉了。若条件不成立，那么 THEN 之后的赋值语句不执行，W的值就是20句赋给的值。所以，第40句输出的W值总是正确的。

至此，有的读者会问，上例中的THEN之后是一个赋值语句，其后能否是IF语句呢？当然是可以的。IF的最典型情况是：

IF〈条件-1〉THEN IF〈条件-2〉THEN…

例4.6 将例中的逻辑条件改为关系条件。

```
10 INPUT "输入数据："；A，X
20 W=COS（A*X+5）
30 IF A<>0 THEN IF X>0 THEN
   W=X/A*EXP（-X*X/(2*A*A))*
     SIN（X）
40 PRINT "A="；A；" "；"X="；X；" "；
                "W="；W
50 PRINT "程序结束"
60 END
RUN↵
输入数据：15，21↵
A=15  X=21  W=.439608606
程序结束
```

这里的第30句只是将前例的逻辑表达式拆成了两个关系表达式，其实际运行效果相同。

4. 多路分支

格式：ON n GOTO 行号-1，行号-2，…，行号-m

作用：根据n的值，选择指定行号中的第n个行号的语句来执行。

这里的n称为条件变量，其值必须是正整数且 $0 \leqslant n \leqslant 255$。n也可以是一个复杂的算术表达式。如果n的值为实数，那么机器自动转换成整数。如果n的值为零或者n值超

界（即n所指的行号不在程序中），那么就执行后继语句。如果n＜0，就发生语法出错。

例如：ON 3 GOTO 20，60，80，因为n＝3，所以应该转到第80句去执行。实际上，n的值必须和行号的排列顺序号相对应。应该特别强调，这种多路分支，不是转到第n（3）行去执行。

例4.7 求Y的值。

$$Y=\begin{cases} A+BX^2+CX^3 & 0.5\leqslant T<1.5 \\ AC\cos BX^2 & 1.5\leqslant T<2.5 \\ \sqrt{A+BX^3}+e^C & 2.5\leqslant T<3.5 \\ \ln\left|B+\dfrac{C}{X}\right| & 3.5\leqslant T<4.5 \end{cases}$$

要用ON形式的条件语句来解决这个问题，关键是构造出一个n，使其值同T有关且刚好落到所需行号上。我们仔细观察就会发现，这里T的变化是有规律的。如果我们直接用T来代表n，那么当T＝0.5时就无法凑成1，因而在T基础上加0.5再取整，即n＝INT（T＋0.5）。试一试能否满足要求。在构造n值的同时，我们假定Y的四种情况依次对应n的1，2，3，4。计算Y的表达式的语句行号在ON中的排列顺序号也要和n值对应。还是先来看下面的程序吧。

```
10 INPUT “输入数据：”；T
20 A＝5：B＝6：C＝7：X＝8
30 N＝INT（T＋0.5）
40 ON N GOTO 60，80，100，120
50 PRINT “T值超界”
55 GOTO 140
```

```
60 Y=A+B*X*X+C*X∧3
70 GOTO 130
80 Y=A*C*COS(B*X*X)
90 GOTO 130
100 Y=SQR(A+B*X∧3)+EXP(C)
110 GOTO 130
120 Y=LOG(ABS(B+C/X))
130 PRINT "T="; T, "Y=" Y
140 PRINT "程序结束"
150 END
RUN↵
输入数据：1↵
T= 1        Y=3973
程序结束
RUN↵
输入数据：2↵
T= 2        Y=26.1813122
程序结束
RUN↵
输入数据：2.5↵
T=2.5       Y=1152.10387
程序结束
RUN↵
输入数据：4↵
T= 4        Y=1.92789164
程序结束
```

这里我们反复运行了四次，每种情况都执行了一次。我

们给出一个T值，就得到相应的一个N值。在上例的第40句中，若

N＝1，相当于GOTO 60；

N＝2，相当于GOTO 80；

N＝3，相当于GOTO 100；

N＝4，相当于GOTO 120。

对于有规律变化的量用ON多路分支语句还是比较方便的。但是，对于无规律或者不容易找出规律的问题，要使用ON就麻烦一些了。对这种情况有两种方法可以解决。一是重复书写ON中的行号，如 ON n GOTO 60，60，80，100，100等。请看下面的例子。另一种实现方法见例4.9。

例4.8 问题同例4.7，只是条件改变了。

$$Y=\begin{cases} A+BX^2+CX^3 & 0\leqslant T<2 \\ AC\cos BX^2 & 2\leqslant T<3 \\ \sqrt{A+BX^3}+e^C & 3\leqslant T<5 \\ \ln\left|B+\dfrac{c}{x}\right| & 5\leqslant T<8 \end{cases}$$

```
10 INPUT "输入数据："；T
20 A＝5：B＝6：C＝7：X＝8
30 N＝INT（T＋1）
40 ON N GOTO 60,60,80,100,100，120，120，120
50 PRINT "T值超界"
55 GOTO 140
60 Y＝A＋B＊X＊X＋C＊X∧3
70 GOTO 130
80 Y＝A＊C＊COS（B＊X＊X）
```

```
90 GOTO 130
100 Y=SQR(A+B*X∧3)+EXP(C)
110 GOTO 130
120 Y=LOG(ABS(B+C/X))
130 PRINT "T=";T,"Y=";Y
140 PRINT "程序结束"
150 END
RUN↙
输入数据：1.5↙
T=1.5      Y=3973
程序结束
RUN↙
输入数据：2.5↙
T=2.5      Y=26.1813122
程序结束
RUN↙
输入数据：3↙
T=3        Y=1152.10387
程序结束
RUN↙
输入数据：5↙
T=5        Y=1.92789164
程序结束
```

第40句中相同的行号60，100，120重复了多次，为的是使条件变量N的值和行号相对应。对于上述问题，也可用下面的办法来实现，

例4.9 用多个IF实现ON的多路分支。

```
10 INPUT "输入数据：" ; T
20 A=5 :B=6 :C=7 :X=8
30 IF T<2  AND T>=0  THEN 60
35 IF T>=2  AND T<3  THEN 80
40 IF T>=3  AND T<5  THEN 100
45 IF T>=5  AND T<8  THEN 120
50 PRINT "T值超界"
55 GOTO 140
60 Y=A+B*X*X+C*X∧3
70 GOTO 130
80 Y=A*C*COS(B*X*X)
90 GOTO 130
100 Y=SQR(A+B*X∧3)+EXP(C)
110 GOTO 130
120 Y=LOG(ABS(B+C/X))
130 PRINT "T=" ; T, "Y=" ; Y
140 PRINT "程序结束"
150 END
RUN↵
输入数据：1.5↵
T=1.5      Y=3973
程序结束
```

请读者自己试一试其它值，运行结果同例4.8是否一样。

实际上，用多个IF来代替ON的功能有时更为简单，因为不必挖空心思去构造条件变量n以及语句行号的对应关系。

不论是转向语句还是条件语句，语句中要求的行号都必须是程序中能够提供的行号，否则就会发生错误。

4.2.2 条件语句的执行流程

前面详细介绍了条件语句的用法，为清楚起见，下面给出它们各自的执行流程图。

1. IF〈条件〉GOTO〈行号〉

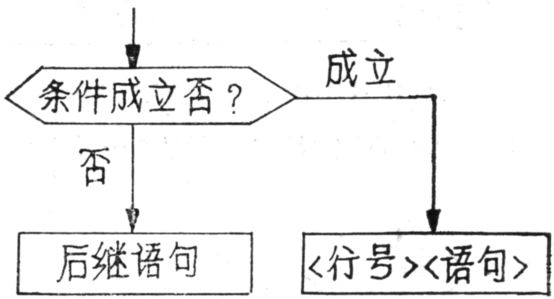


图4.1 IF-GOTO执行流程图

2. IF〈条件〉THEN〈行号〉

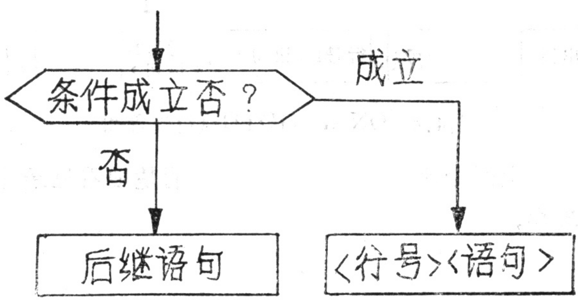


图4.2 IF-THEN执行流程图

3. IF〈条件〉THEN〈语句序列〉

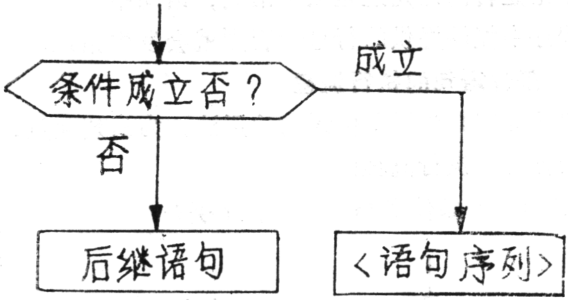


图4.3　IF-THEN〈语句〉执行流程图

4．　ON n GOTO《行号1，行号2…行号m》

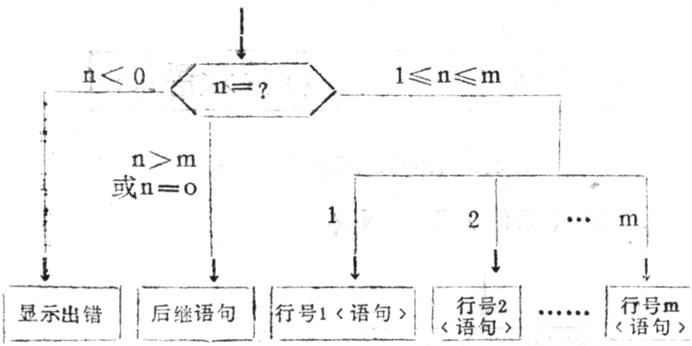


图4.4　ON n　GOTO执行流程图

从以上四个流程图，我们就可以清楚地看见条件语句的执行流程。

4.3　读数据和置数据语句

由前面的示例中已经发现，为使变量获得数据，几乎都

采用INPUT语句。虽然这种输入数据的方式来得简单和方便，但如果我们要重新执行一次程序，那么仍然得重新从键盘上敲入值，即使先前已经输入过的数据，也必须如此。有什么办法能使我们仅敲入一次数据而可以多次重新运行程序呢？读数据和置数据语句是最好的帮手。

4.3.1 读数据和置数据的格式和作用

读语句格式：READ 〈变量表〉

作用：将DATA提供在数据区的数据依次送入到指定变量名中。

这里的变量表是一个或多个变量名，如果有多个变量名，变量名彼此之间要用“，”隔开。

置数据语句格式：DATA 〈数据表〉

作用：为机器内部开辟的数据区提供数据。

数据表是一个或多个数据。如果是多个数据，数据彼此之间必须用“，”隔开。这里所说的数据只能是数值常数或字符串常数，不能是变量名、函数或带运算符的表达式。

如果一个程序中使用了READ语句，那么程序中一定要有DATA语句。

我们举例说明READ和DATA的用法。

例4.10 编写一个程序，它能够按如下形式统计和打印学生成绩。

学号 (X_1)	姓名 ($X_2\$$)	数学 (X_3)	语文 (X_4)	物理 (X_5)	化学 (X_6)	总分 (W)
631	张扬	85	90	92	73	…
633	李力	93	86	90	91	…
635	王可为	78	82	80	76	…
平均成绩		…	…	…	…	…

假设：X_1——学号，$X_2\$$——姓名，X_3——数学，X_4——语文，X_5——物理，X_6——化学，W——总分。S_3——数学的和，S_4——语文的和，S_5——物理的和，S_6——化学的和，S_7——总分和，i——学生人数。

解决该问题的思路是：

①读入一个学生的学号、姓名以及各科成绩。

②将各单科成绩累加求和。

③求各个学生的各科总分。

④用DATA提供数据，用READ读入数据。

⑤按报表格式打印。

下面是程序清单：

```
5  S3=0:S4=0:S5=0:S6=0:S7=0
10 I=1
15 PRINT "学号"; "姓名"; "数学"; "语文";
        "物理"; "化学"; "总分"
20 W=0
25 I=I+1
30 READ X1, X2$, X3, X4, X5, X6
40 W=X3+X4+X5+X6
50 S3=S3+X3
60 S4=S4+X4
70 S5=S5+X5
80 S6=S6+X6
90 S7=S7+W
100 PRINT X1; " "; X2$; " "; X3; " ";
        X4; " "; X5; " "; X6; " "; W
110 IF I<>3 THEN 20
```

```
120 PRINT "平均成绩";S3/I; " ";S4/I; " ";
S5/I; " "; S6/I; " "; S7/I
130 DATA 631, "张扬", 85, 90, 92, 73
135 DATA 633, "李力", 93, 86, 90, 91
140 DATA 635, "王可为", 78, 82, 80, 76
150 END
RUN↵(以下是运行的结果)
学号  姓名    数学   语文  物理   化学  总分
631   张扬    85     90    92     73    340
633   李力    93     86    90     91    360
635   王可为  78     82    80     76    316
平均成绩      85.33  86    87.33  80    338.66
```

这个程序中用到了READ和DATA语句。DATA的作用就是把张扬、李力、王可为三个学生的学号、姓名和成绩依次放在数据区中，当执行第30句的READ语句时，就直接到数据区去取学生的这些数据。经过统计和计算处理，打印出报表。

READ和DATA究竟是如何配合使用的呢？原来，机器在内存中开辟了一个区域，专门用来存放DATA提供的数据，称为数据区。按照DATA在程序中出现的先后顺序，依次将DATA提供的数据存放在数据区中。为了指出数据区中数据的位置，系统内部还专门设立了一个指针，指向数据区的第一个数据位置，我们称为初始位置。随着数据的读取，指针可以自动向后推移。指针总是指向正要读取的数据。在本例中，数据和指针的初始位置如图4.5所示。

当输入程序后，DATA提供的数据在数据区内存放的情况如图4.5(a)所示，指针指向当前值631。这就是说，

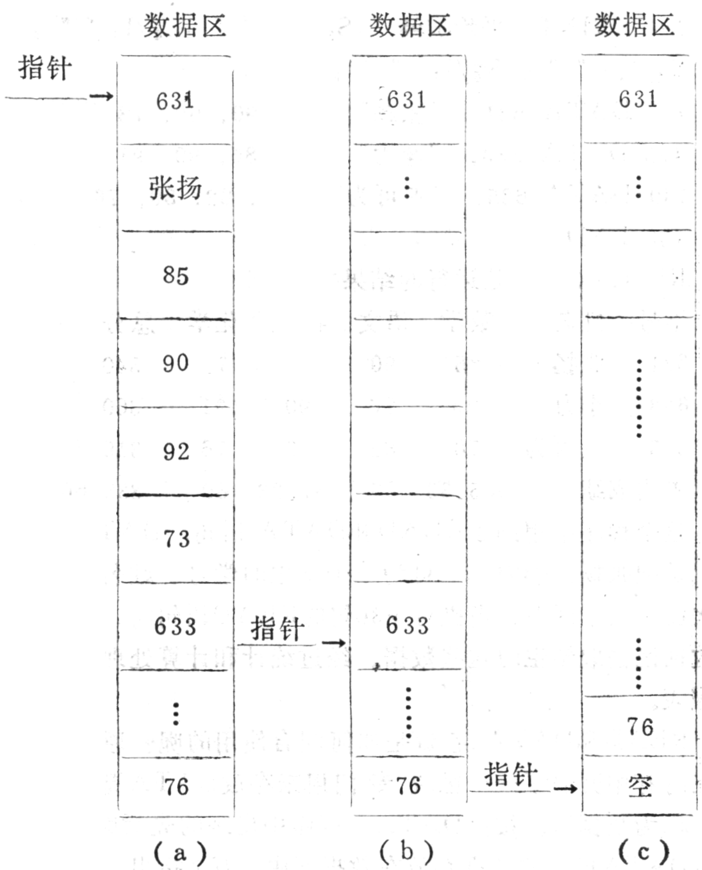


图4.5 READ和DATA执行示意图

如果要读入的话，首先读入的就是631。当第30句执行以后，就把张扬的数据取到了相应的变量中。这时指针已指向633这个值了（准备下一次读取）。如图4.5（b）所示。第二次执行READ时，李力的数据又读入到相应的变量中，指针指向635。如此重复，直到把三个学生的数据读完。当读取

了王可为的化学成绩76以后，指针就指向数据区末尾的一个空单元，如图4.5（c）所示。如果这时继续执行READ语句，就会出现数据不够用的错误。因为指针指向空单元就表示已经没有数据可供读取了。数据区和数据指针把READ和DATA语句联系起来了。

从以上可知，READ并不是直接在DATA语句位置去读取数据，因而DATA在语句中的位置是无所谓的，可在READ之前，也可在其后。

READ和DATA语句的优点是当我们再次用RUN重新运行程序时不必再输入数据，因为数据已写入了程序中，而每次用RUN重新运行时，系统总是把指针指向初始位置。缺点也是显然的。由于数据均要写入程序中，因而增加了程序的长度。

4.3.2 使用READ和DATA时注意

1．READ和DATA必须联合使用

只要程序中用到了READ语句，不管用了几次，也不管书写了几个，至少应有一个DATA语句与之对应。由于READ只能从数据区读取数据，因而必须有DATA向数据区提供数据，也只有DATA才能向数据区提供数据。

2．READ中变量的个数、类型、顺序必须和DATA提供的数据相对应。

一般来说，READ中有几个变量名，DATA就必须提供几个数据（当然可以多于变量的个数）。如果变量名多而数据少，则有些变量就得不到值。因而发生读入数据出错。

3．一个程序中可以有一个很长的DATA语句（即数据多），也可以把一个长的DATA拆成几个短的DATA(即

数据少一些）。如例4.10中的130、135、140语句可以合并成一个DATA，也可以把它们再分小一些。通常把相对独立的数据放在同一个DATA之后，便于检查和修改。如例4.10那样，每个学生的数据用一个DATA语句。DATA的个数不影响数据区。

4．DATA之后只能是常数。包括数值常数（可正可负）和字符串常数，且数据之间必须用“，”隔开。

5．DATA在程序中的位置是任意的。但不同行号的两个DATA的先后顺序是重要的，因为它们的先后顺序决定了其提供的数据在数据区内的先后顺序。同样的，同一个DATA之后的数据排列的先后也是重要的，这个顺序取决于它和READ中变量名的对应关系。

4.4 数据指针的恢复

我们已经知道，每从数据区读取一个数据，数据指针就向下移动一个位置。当把数据区中数据读完以后，指针就指向数据区末尾的一个空单元，这时如果继续读数据就会发生错误。如果我们再从头开始读取数据呢？当然是可以的。但必须设法把指针拨回到开始位置。所谓数据指针的恢复，就是把数据指针拨回到数据区的开始位置，使之指向第一个数据。如图4.5中的（c），指针已经在数据区的末尾了，恢复指针就是把它重新指向第一个数据631，恢复到图4.5（a）的情况。恢复数据指针用到的语句是RESTORE。

语句格式：RESTORE

作用：将数据区中的数据指针拨回到数据区的初始位置（指向第一个数据），以便READ语句重新读取数据区的数

据。

例4.11　求下式的值。

$$Y=\sqrt{A_1+A_2+A_3+A_4+A_5}+\sqrt[3]{B_1+B_2+B_3+B_4+B_5}+\sqrt[4]{C_1+C_2+C_3+C_4+C_5}$$

其中：A_1，A_2，A_3，A_4，A_5 分别取值为11.42，135，72.4，58.3，66.4

B_1，B_2，B_3，B_4，B_5 分别取值为：11.42，135，72.4，58.3，66.4

C_1，C_2，C_3，C_4，C_5 分别取值为：11.42，135，72.4，58.3，66.4

显然，三组值对应相同。对这个问题的程序可以用多种办法来实现。

解法 1　直接读入数据进行计算。

```
10 READ A1, A2, A3, A4, A5
20 READ B1, B2, B3, B4, B5
30 READ C1, C2, C3, C4, C5
40 W1=A1+A2+3+A4+A5
44 W2=B1+B2+B3+B4+B5
48 W3=C1+C2+C3+C4+C5
50 Y=SQR(W1)+W2∧(1/3)+W3∧(1/4)
60 PRINT "Y=" ; Y
70 DATA 11.42, 135, 72.4, 58.3, 66.4
74 DATA 11.42, 135, 72.4, 58.3, 66.4
78 DATA 11.42, 135, 72.4, 58.3, 66.4
```

```
80 PRINT "程序结束"
90 END
RUN↵
Y=29.8429754
程序结束
```

程序是正确的，但不是理想的。因为同样的数据重复书写和存贮了三次。这不但增加了书写麻烦，更重要的是浪费了存贮空间。如果使用 RESTORE 语句，数据就简短多了。

解法 2　用RESTORE来恢复数据区。

```
10 READ A1, A2, A3, A4, A5
15 RESTORE
20 READ B1, B2, B3, B4, B5
25 RESTORE
30 READ C1, C2, C3, C4, C5
40 W1=A1+A2+A3+A4+A5
44 W2=B1+B2+B3+B4+B5
48 W3=C1+C2+C3+C4+C5
50 Y=SQR(W1)+W2∧(1/3)+W3∧(1/4)
60 PRINT "Y="; Y
70 DATA 11.42, 135, 72.4, 58.3, 66.4
80 PRINT "程序结束"
90 END
RUN↵
Y=29.8429754
程序结束
```

程序的执行情况是：当10句读入A_1、A_2、A_3、A_4、

A_5以后，指针已指向数据 区 末尾空白单元了。通过执行第15句的RESTORE语句，就把 指针拨回到第 1 个数据11.42的位置。当第20句读入B_1、B_2、B_3、B_4、B_5时，指针又依次下推。如此重复。巧妙的使用RESTORE语句，就可以使我们能够重复读取数据区的数据。对于这个特定的问题，还有更简洁的程序如下：

解法 3　因为三组数据对应相等，因而只需读入一组值就可以完成计算了。

```
10 READ A1, A2, A3, A4, A5
20 W=A1+A2+A3+A4+A5
50 Y=SQR(W)+W∧(1/3)+W∧(1/4)
60 PRINT "Y=";Y
70 DATA 11.42, 135, 72.4, 58.3, 66.4
80 PRINT "程序结束"
90 END
RUN↲
Y=29.8429754
程序结束
```

在使用RESTORE时需要注意：

1. RESTORE只能把数据指针拨回到数据区的初始位置，而不能拨到中间某个任意的数据位置。

2. 只有当需要重头读取数据时才需要 用 RESTORE来恢复指针。

3. 哪个READ语句需要从数据区的第一个数据位置重新开始读入，就在那个READ语句 执行之前给出一个RESTORE语句（在程序中RESTORE 的 顺序不一定必须放在READ之前，但其执行必须在READ之前）。

4. 如果恢复指针后某些位置上的数据并不需用，可以用无用的变量名占据相应的数据位置。如我们仅需要恢复B_3、B_4、B_5和C_3、C_4、C_5，可用如下程序：

例4.12 恢复读取部份数据。

```
10 READ A1, A2, A3, A4, A5
15 RESTORE
20 READ T, T, B3, B4, B5
25 RESTORE
30 READ T, T, C3, C4, C5
40 PRINT A1, A2, A3, A4, A5
50 PRINT B3, B4, B5, C3, C4, C5
60 PRINT "程序结束"
70 DATA 11.42, 135, 72.4, 58.3, 66.4
80 END
RUN↵
11.42     135     72.4     58.3     66.4     (A的值)
72.4      58.3    66.4     (B的值)
72.4      58.3    66.4     (C的值)
```

这里的T就是无用变量，它纯碎就是用来占据11.42和135两个数据位置的。

5. 需要特别强调的是，RESTORE不是使控制返回到程序中的第一个READ语句去重新读入数据。RESTORE只管恢复数据指针，至于是否用READ去读入，则由用户根据问题需要来确定。

4.5 框图法

在程序设计中，如果问题比较简单，那么可以直接编写

程序。但是，实际应用中的问题往往比较复杂，需要考虑的因素比较多，程序的逻辑结构也很复杂，因而面对问题直接编写程序就会觉得千头万绪无从下手。为此，我们引入程序设计中的框图法。所谓框图法，就是把解决问题的步骤用方框图的形式表示出来，然后再根据方框图来编写程序。一般说来，解决问题的框图画出来以后，根据框图来编写程序就不困难了。

1．框图法的表示符号

①用菱形框或半圆框来表示判断条件，框内写出条件名称。如：〈条件成立否？〉或（条件成立否？）

②用长方框来表示需要进行的操作，框内写出操作名称。如：［A⇒B］

③用箭头标出逻辑流向。通常用箭头把条件框和操作框联系起来。如图4.6所示。

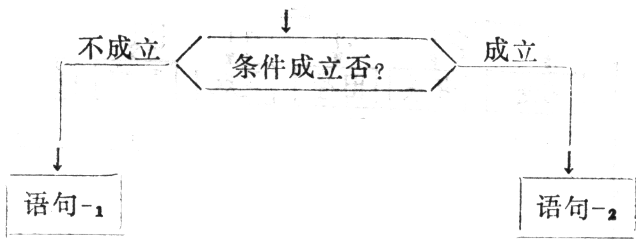


图4.6　框图联系示意图

一个实际问题不论有多复杂，均可用这种框图描述出来。根据用户的水平和需要，框图画得可粗（大概步骤）可细（画出详细步骤）。

2．框图的例子

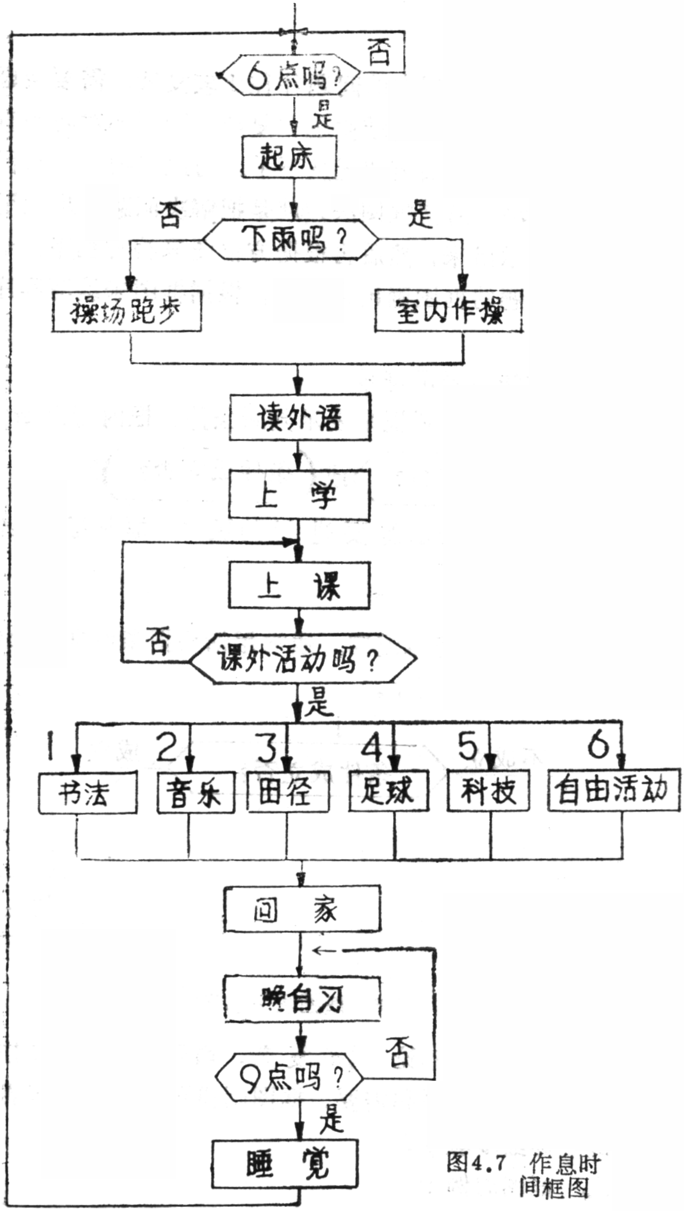

6点吗？
否
是
起床
下雨吗？
否
是
操场跑步
室内作操
读外语
上　学
上　课
课外活动吗？
否
是
1
2
3
4
5
6
书法
音乐
田径
足球
科技
自由活动
回　家
晚自习
9点吗？
否
是
睡　觉

图4.7　作息时间框图

实际上，我们每个人每天都在按一定的框图生活、工作和学习。

例4.13　学生的作息时间框图。

图4.7的框图还是比较粗的，它只描绘了一天的活动轮廓。我们还可以把每天的作息时间详细画出来。参照这个例子，请读者自己完成。

4.6　程序实例

例4.14　求一个最小的正整数，它满足如下的条件：除2余1，除3余2，除4余3，除5余4，除6余5。

我们可以这样来考虑问题：根据所给条件，列出可能的正整数有：

除2余1：3，5，7，9，11，…

除3余2：5，8，11，14，17，…

除4余3：7，11，15，19，23，…

除5余4：9，14，19，24，29，…

除6余5：11，17，23，29，…

我们要找的那个正整数，肯定是所有数列中均有的那个数。从上面列出的正整数可以看出，要使所有条件均满足，首先必须满足除6余5这个条件。因为该条件生成的数才有可能满足其余条件。因此，我们用除6余5所生成的数进行试探。如果满足条件的数是存在的话，那么经过有限次的试探后就一定能够找出这个数。从上面列出的数列可以看到，除6余5的最小数是11，而后面每个数都比前一个数多6。这就是说，用前一个数测试不行的话，就再增加6继续测试。如此下去，总可以找出满足所有条件的那个最小正整

数。下面画出解决该问题的框图。假设待测试的那个数是X。

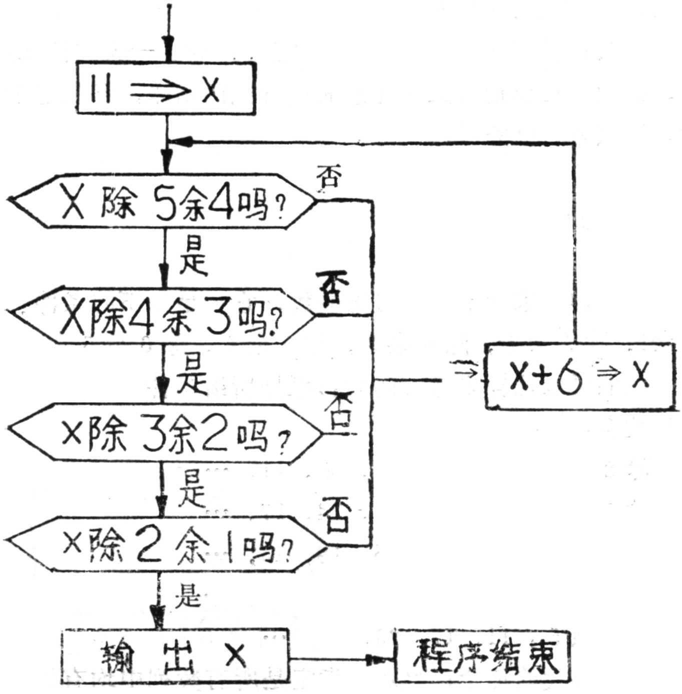


图4.8　例4.14的框图

下面是程序清单：

```
10 X=11
20 IF X=INT(X/5)*5+4 AND
      X=INT(X/4)*4+3 AND
      X=INT(X/3)*3+2 AND
      X=INT(X/2)*2+1 THEN PRINT
```

```
        "X=" ; X; GOTO 50
30 X=X+6
40 GOTO 20
50 PRINT "程序结束"
60 END
RUN↙
X=59
程序结束
```

说明：

①X=INT（X/5）＊5＋4刻划了除5余4这个条件。其余几个条件的刻划类似。

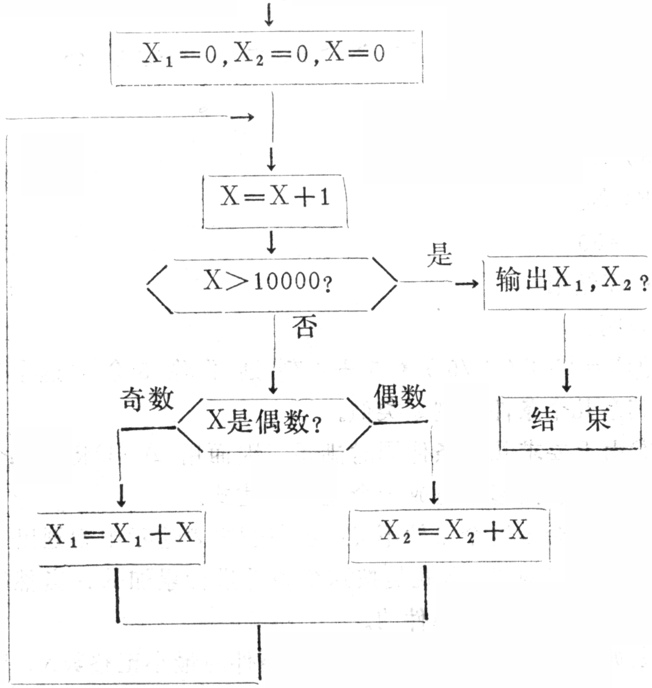
②由于要求几个条件同时满足，因而用AND将这几个关系表达式连接起来组成一个逻辑表达式。

③除6余5这个条件不必再在逻辑表达示中刻划出来了，因为我们构造的X就是按这个条件进行累加的，当然它是满足除6余5这个条件的。

④如果根本就不存在满足所给条件的最小正整数X，那么我们就在第30句之后对X给一个限制条件。当X的值超过某个足够大的数以后，我们就认为所求的数不存在了。请读者自己补上这个限制条件。

例4.15　求10000以内所有奇数之和、偶数之和。

假设用X_1代表奇数和，X_2代表偶数和，X代表1～10000之间的自然数。




```
10 X1= 0 :X2= 0 :X= 0
20 X=X+1                  (形成1,2,3,…,10000)
25 PRINT X
30 IF X>10000 GOTO 90
40 IF X=INT(X/2)*2  THEN 70
50 X1=X1+X                 (奇数累加)
60 GOTO 20
70 X2=X2+X                 (偶数累加)
80 GOTO 20
```

```
90 PRINT "奇数之和="；X1，"偶数之和="；X2
100 PRINT "程序结束"
110 END
RUN↵
奇数之和=25000000      偶数之和=25005000
程序结束
```

由于程序运行时间较长，为了不致太寂莫，第25句不断的输出1，2，3…10000的自然数。

例4.16 编写一个程序，它能够统计学生成绩并按分数段打印下列表格：

分数段	100	90～99	80～89	70～79	60～69	60以下
人数	…	…	…	…	…	…
百分比%	…	…	…	…	…	…

假设分别用G_{100}、G_{90}、G_{80}、G_{70}、G_{60}、G_{59}代表各分数段的人数，用G来代表学生的成绩。X_{100}、X_{90}、X_{80}、X_{70}、X_{60}、X_{59}代表各分数段所占人数百分比。I代表学生人数。解题思路是：

①读入一个成绩，就判断其所属分数段。

②求各分数段所占人数百分比。

③按表格要求输出。

这个问题比较简单，我们直接给出程序：

```
5  I=0:G100=0:G90=0:G80=0:G70=0:
   G60=0:G59=0
10 PRINT "读入学生成绩："
15 READ G
20 IF G=-1 THEN 90
25 I=I+1
```

```
30 IF G=100 THEN G100=G100+1:GOTO 10
40 IFG>=90 AND G<99 THEN G90=G90+1:
   GOTO 10
50 IF G>=80 AND G<89 THEN G80=G80+1:
   GOTO 10
60 IF G>=70 AND G<79 THEN G70=G70+1:
   GOTO 10
70 IF G>=60 AND G<69 THEN G60=G60+1:
   GOTO 10
80 G59=G59+1:GOTO 10
90 X100=G100/I*100:X90=G90/I*100:X80=
   G80/I*100
95 X70=G70/I*100:X60=G60/I*100:X59=G59/I
   *100
100 PRINT"分数段";"100 90～99 80～89 70～
    79 60～69 60以下"
110 PRINT"人数";G100;" ";G90;" ";
    G80;" ";G70;" ";G60;" ";G59
120 PRINT"百分比%";X100;" ";X90;
    " ";X80;" ";X70;" ";X60;" ";
    X59
130 DATA 80, 85, 91, 93, 76, 84, 66, 79, 74,
    78, 63, 62, 88, 100, 42
135 DATA 74, 86, 82, 78, 95, 77, 84, 66, 78,
    94, 85, 67, 53, -1
140 END
RUN↵
```

分段数	100	90～99	80～89	70～79	60～69	60以下
人数	1	4	8	8	5	2
百分比%	3.57	14.285	28.57	28.57	17.857	7.14

说明：

①第135句末尾的-1不是学生成绩，而是人为安排的数据结束的标志，它用于第20句判断读入数据是否完结。数据读入结束标志可以是任何一个不同于DATA提供的其它数据。

②为使打印的表格美观整齐，在安排打印表格时，有关行列应对整齐，以免输出时数据错位。

③为使变量名和分数段相对应 本例中用了G_{100}，X_{100}等作变量名。构成变量名的字符超过了两个，但由于所有变量名的第二个字符均不相同，所以不会发生混乱。

4.7 源程序如何打印和存盘

源程序输入机内，经过调试、修改、运行，证明程序已经正确无误。如何将源程序清单打印出来？又如何将源程序保存在磁盘上呢？

打印源程序清单，CEC-I机上分中文和西文两种方式。

1. 西文方式下打印程序清单

首先用PR#1命令连通打印机，再用 LIST列出源程序就可以了。屏幕显示如下：

] PR# 1 ↵　　　（连通打印机）

] LIST ↵　　　（将源程序在宽行纸上打出）

如果不再打印，可用 PR# 0 断开打印机。如：] PR# 0 ↵ 以后再用LIST时就只能在屏幕上列出源程序了。

2．中文方式下打印程序清单

用 POKE 1659，〈字型号〉连通打印机，再用 LIST 列出程序清单，这个操作在屏幕上显示如下：

] POKE 1659，2 ↵ （表示用 2 号字体）

] LIST ↵ （将源程序打印在宽行纸上）

如果打印中需要不同型号的汉字，就可在〈字型号〉中给出相应的号码。CEC－Ⅰ可提供15种型号的汉字，其字型编号分别为1-15。详细情况请参见第十三章。

3．源程序如何存盘和读入

源程序输入、修改、调试、运行以后，除了可用前面介绍的方法打印在纸上以外，最好保存在磁盘上，以备随时调用。下面介绍两个与BASIC源程序存盘有关的命令。

格式：SAVE〈文件名〉

作用：将内存中的BASIC源程序用文件的形式保存到磁盘上。

这里的文件名，就是给BASIC源程序取的名。它可以是字母和数字字符。文件名最好能够暗示源程序的用途或意义。

格式：LOAD 〈文件名〉

作用：将文件名所代表的源程序从磁盘上调入内存，从而可以对源程序重新调试、修改或运行。

这里的文件名就是先前用SAVE存入的源程序的名字。我们用一个例子来说明。

例4.17 将例4.16的源程序保存到磁盘上，然后再调入内存运行之。我们用E4-16代表4.16的源程序。这意思是第四章的第16个例子。这个操作在屏幕上显示为：

] SAVE E4-16 ↵ （把E4-16保存到磁盘上）

```
] LOAD E4-16↵          （调入E4-16到内存）
] RUN↵                 （运行程序）
```

（打印结果略去了）

问题与思考

1．ON n GOTO 行号-1，行号-2，…，行号-m形式的语句不是必需的，为什么？

2．为什么DATA语句可以放在程序的任何位置？

3．举一个RESTORE语句放在READ语句之后而又能达到恢复目的的例子。

习　题

1．指出下列程序运行的结果，并指出数据指针处在什么位置。

①
```
10 READ A, B, C
20 A=A+B+C
30 B=A+B+C
40 C=A+B+C
50 IF C>A+B+C THEN 20
60 PRINT "A="; A, "B="; B, "C="; C
70 DATA 1, 2, 3
80 END
RUN↵
```

②
```
10 READ A, B, C, D, E, F
20 DATA 1, 2, 3
30 A=A+E+F
40 DATA 4, 5, 6
50 G=G+B+C+D
```

```
60 DATA 7, 8, 9, 10, 11, 12, 13
70 RESTORE
80 IF G<30 THEN 10
90 PRINT A, B, C, D, E, F, G
100 END
RUN↵
```

③
```
10 READ A, B, C
20 A=A+B+C
30 IF A>B THEN 50
40 B=A+B+C
50 IF B>C THEN 80
60 C=A+B+C
65 B=A+B+C
70 IF C>A THEN 20
80 PRINT A, B, C
90 DATA 10, 20, 30
100 END
RUN↵
```

2. 指出下列程序中的书写错误。

①
```
10 READ A, B, C, D
20 W=A+B+C+D
30 PRINT W, A, B, C, D
40 DATA 2, 4/5, -8
50 END
```

②
```
10 INPUT A, B, C
20 W=A+B+C
30 IF W>30 THEN 100
```

```
40 PRINT W, A, B, C
50 END
RUN ↵
? 10, 20, 30
```

3. 举一个现实生活中框图的例子并画出来。

4. 编写下式的程序。

① $$F=\begin{cases} X^5 & X\leqslant 0 \\ (1-X)^5+1 & 0<X<1 \\ X^5+1 & X\geqslant 1 \end{cases}$$

② $$f(X,Y)=\begin{cases} \sqrt{X^2+Y^2} & X<Y \\ e^{XY} & X=Y \\ \dfrac{X-Y}{XY} & X>Y \end{cases}$$

③ $$F(X,Y)=\begin{cases} X^2+Y^2+2XY & X\leqslant 0,\ Y\leqslant 0 \\ XY-X^2-Y^2+3X & X>0,\ Y\neq 0 \\ 2XY+X^2-7Y & X>0,\ Y>0 \end{cases}$$

5. 先画出框图，再编写出程序。

① 先用ON多路分支语句来实现。

② 再用多个IF语句来实现。

$$F(t)=\begin{cases} \sqrt{t^2+1} & 0<t<1 \\ t^2-1 & 1\leqslant t<2 \\ t^2-2t+1 & 2\leqslant t<3 \\ t^2-4t-25 & 3\leqslant t<4 \end{cases}$$

6. 编写程序。

① 求1＋2＋3＋…＋100的和。

② 求1×2×3×…×50的积。

7. 先画框图再编写程序。

$$i=\begin{cases}\dfrac{V}{WL}e^{\alpha t}\sin wt & R^2-\dfrac{4L}{C}<0\\ \dfrac{V}{L}te^{\alpha t} & R^2-\dfrac{4L}{C}=0\\ k_1e^{s_1t}+k_2e^{s_2t} & R^2-\dfrac{4L}{C}>0\end{cases}$$

其中：$W=\sqrt{\dfrac{1}{LC}-\dfrac{R^2}{4L^2}}$， $\alpha=\dfrac{-R}{2L}$

$$K_1=\frac{-V}{2L\sqrt{\dfrac{R^2}{4L^2}-\dfrac{1}{LC}}},\quad k_2=-k_1$$

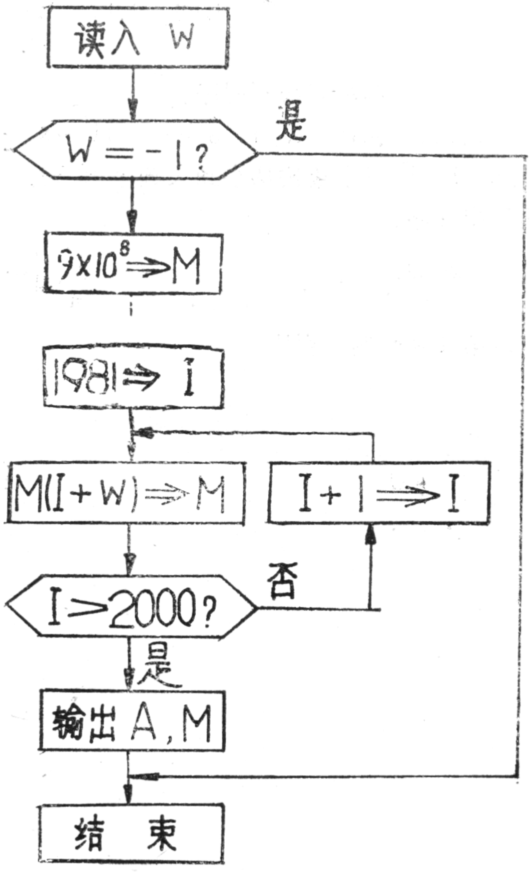
$$S_1=-\frac{R}{2L}-\sqrt{\frac{R^2}{4L^2}-\frac{1}{LC}},$$

$$S_2=-\frac{R}{2L}+\sqrt{\frac{R^2}{4L^2}-\frac{1}{LC}}$$

假定R、L、C、V为已知常数，求t＝0，0.1，0.2，…10时i的值各是多少。

8. 根据下列程序画出框图。

```
5 W=100
10 READ X
15 IF X=999 THEN 70
20 W1=W
25 Y=0
30 W1=W1*(1+X)
```




```
35 Y=Y+1
40 IF W1>2*W THEN 50
45 GOTO 30
50 PRINT "X="; X, "Y="; Y
55 GOTO 10
```

```
60 DATA 0.05, 0.07, 0.09, 0.11, 0.13, 999
70 END
```

9. 请根据123页框图来编写程序。

W＝0.02，0.015，0.01，5E－3，1E－3，－1。

10. 将a，b，c 三个数按从小到大顺序排列。先画出框图，再写出程序。

11. 已知数列：$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \cdots, \frac{1}{n}, \cdots$。问小于$10^{-8}$的是第几项？写出程序。

第五章　循环程序

在现实生活中，循环的例子实在是太多了。如象我们生活的昨天、今天和明天，一天复一天，真可谓“明日复明日，明日何其多”。昨天、今天和明天，似乎天天都在重复，而天天都不完全一样。去年，今年，明年，年复一年，永远如此。

5.1　什么是循环？

在BASIC语言中，也有专门用于描述“明日复明日”的语句，这就是BASIC的循环语句。在程序设计中，经常需要反复执行一段程序，而在循环的执行过程中，程序的执行情况也在悄悄地发生变化。如象日复一日，年复一年，表面上看起来没有什么变化，但从人的年龄、知识、对财物的消耗来说都在发生变化一样。很难想象，生活中如果没有循环，世界将会变成什么样子。同样，如果BASIC语言（其它语言也如此）不允许或不能解决循环问题，那么很多问题就只能是可想而知了。例如，求1＋2＋3＋…＋100000的值，我们可以借助99999个加号“＋”将这些数字加起来，但实际上是行不通的。

事实上，BASIC语言有两种方法可以用来解决循环问题。一是条件语句结合转向语句；二是用BASIC提供的FOR-NEXT循环语句。

5.2　循环语句

5.2.1　循环语句的格式和作用

格式：FOR〈实型算术变量〉=〈算术表达式-1〉TO
〈算术表达式-2〉〔STEP〈算术表达式-3〉〕
〈语句序列〉
NEXT〈实型算术变量〉

作用：反复执行FOR和NEXT之间的语句序列，直到实型算术变量的值超过算术表达式-2的值为止，然后再顺序执行NEXT的后继语句。

其中：FOR—循环的开始标志。〈实型算术变量〉——一个取实数值的变量名，又称循环控制变量。循环控制变量的值可以决定循环是否继续进行。

〈算术表达式-1〉—又称为循环控制变量的初值，即第一次循环时的取值。它可以是一个数值常数，也可以是一个带有各种算术运算符的复杂算术表达式。循环执行时，机器首先将初值计算出来，再赋给循环控制变量，作为第一次循环的开始值。

〈算术表达式-2〉——用于控制循环终止的，又称循环终限。如果循环控制变量的值超过了这个终限值，那么就停止循环而顺序执行NEXT的后继语句。如果循环控制变量的值没有超过终限值，则继续执行FOR和NEXT之间（即循环体）的语句序列。终限可以是一个具体的数值常数，也可以是一个算术表达式，但循环控制变量和终限进行超过比较时，总是按代数值来比较。

TO—终限的标志，即说明TO之后紧跟的是循环语句的终限。

〈算术表达式-3〉——称为循环控制变量的步长或增量，即表示每次循环后循环控制变量又要增加多少值后才又进行下一次循环。步长同样可以是一个具体的数值，也可以

是一个算术表达式。步长可以是正数，也可以是负数。STEP是步长的标志。〔STEP〈算术表达式-3〉〕是一个待选项，根据需要进行取舍。如果步长为1，则待选项可以不写。暗示循环控制变量每次递增1。

〈语句序列〉——又称循环体，即需要反复执行的部份。它由一个或多个语句组成。当然，循环体中可以包含赋值语句、条件语句、转向语、输入语句、打印语句，甚至还可以是循环语句。

NEXT——循环范围的结束标志。FOR和NEXT构成了需要循环执行的部份。FOR之前和NEXT之后均不属该循环的范围。通常，系统要求在NEXT之后的同行上紧跟一个循环控制变量名，以标志NEXT是哪一个循环语句的结束标志。

我们还是来看看下面的例子吧。

例5.1　求100以内所有奇数的和。

```
10  S=0
20  FOR  I=1  TO  100  STEP  2
30          S=S+I
40  NEXT  I
50  PRINT "S=" ; S
60  END
RUN ↵
S=2500
```

在这个具体例子中，第20句中的FOR是循环的开始标志，I是循环控制变量，1是I的初值，100是I的终限，2是步长。第30句的S=S+I是循环体，这里循环体仅有一个赋值语句。NEXT I表示循环范围到此为止。从20～40句一

起构成BASIC中的一个循环语句。

这个循环语句的执行情况是：

①将初值1送到循环控制变量I中；

②执行循环体S=S+I，即S=S+1。

③将循环控制变量增加一个步长（2），即I=I+2，将I同终限值100比较，若I超过（这里暂时理解为大于）终限值，就跳出循环后转④。否则，再转到②继续执行循环体。

④执行NEXT I的后继语句，即PRINT "S="；S。

显然，上例完成了1+3+5+7+…+99的累加计算，结果存放在S中。

例5.2 用循环语句编写求10000以内奇数之和、偶数之和的程序。

我们仍然用X_1累加奇数，X_2累加偶数。

```
10  X1=0：X2=0
20  FOR  X=1  TO  10000
30  IF  X=INT(X/2)*2  THEN  60
40  X1=X1+X
50  GOTO  70
60  X2=X2+X
70  NEXT  X
80  PRINT "奇数之和="；X1，"偶数之和="；
      X2
90  END
RUN ↵
奇数之和=25000000      偶数之和=25005000
```

这个循环语句从第20句开始，到第70句结束。30～60句是循环体。由于该循环语句的步长是1，因而把STEP 1省

略不写了，这是允许的。但是要注意，只有步长为1时才能省写，其它情况都不能省写。

例5.3 求10000以内的奇数之和、偶数之和也可用两个循环语句来实现。

```
10  X1＝0：X2＝0
20  FOR  I＝1  TO  9999  STEP  2
30  X1＝X1＋I
40  NEXT  I
50  FOR  J＝2  TO  10000  STEP  2
60  X2＝X2＋J
70  NEXT  J
80  PRINT"奇数之和＝"；X1，偶数之和＝"；
    X2
90  END
RUN ↙
奇数之和＝25000000   偶数之和＝25005000
```

从运算结果来看完全相同。20～40句求奇数之和，50～70句求偶数之和。奇数和偶数的形成是由循环语句自动生成的。从简洁和运行速度来说，例5.3优于例5.2。请读者用手工方法来跟踪几次循环，以体会循环的执行情况。

例5.4 设A＝3，B＝5，X＝0.5，步长为1.5，直到10时，求

$$Y=\frac{AX^2+BCOSX}{AX^2+BX^3SINX}$$ 的值

```
10  A＝3：B＝5：I＝0
20  FOR X＝0.5  TO  10  STEP  1.5
```

```
25  I=I+1
30  W=A*X*X
40  Y=(W+B*COS(X))/(W+B*X∧3*SIN
    (X))
50  PRINT "I=";I,"X=";X,"Y=";Y
60  NEXT X
70  PRINT "程序结束"
80  END
RUN ↵
I= 1         X=0.5        Y=4.89492407
I= 2         X= 2         Y=.205062576
I= 3         X=3.5        Y=-.834029171
I= 4         X= 5         Y=-.145745333
I= 5         X=6.5        Y=.311825431
I= 6         X= 8         Y=.0701980001
I= 7         X=9.5        Y=-5.16915494
程序结束
```

在程序中，变量I不但用于统计X、Y的组数，也可以用于记录循环的次数。

仔细观察输出的数据，读者就会发现，X取值9.5以后，程序就停止循环了，为什么没有取到终限值10呢？因为9.5+1.5=11，这个值已经超过了终限10，因而跳出循环。由此可以得出一般结论：循环语句中循环控制变量的值不一定刚好取到终限值。我们为什么称为终限而不称为终值就是这个原因。

5.2.2 循环语句的执行流程

在不同型号机器上的循环语句，其循环效果是相同的。但循环语句在具体实现时，执行流程可能不相同。对循环的处理来说，有的机器是循环控制变量得到初值后就马上与终限值进行比较，如果初值一开始就超过终限，那么导致循环一次都不执行。而有的机器又是循环控制变量得到初值后立即执行一次循环，然后将步长累加到循环控制变量中与终限值进行比较，以确定是否进行下一次循环。这种处理方法导致循环至少要执行一次，而不管初值和终限值如何。前一种方法我们称为是先比较后确定循环，后一种方法称为先循环后比较。处理方法不同，得到循环的结果就不一样。中华学习机CEC-I采用的是先执行后比较的循环处理方法，因而，不管循环控制变量的初值和终限如何，总是要执行一次循环的，是否跳出循环，只有第二次循环开始时才能确定。

下面给出处理循环语句的执行流程图5.1。

"循环控制变量超过终限否"中所谓"超过"的含义取决于步长的取值符号。

如果步长是正值（即增量），那么"超过"的意思就是循环控制变量的值大于终限值；如果步长是负值（即减量），那么"超过"的意思就是循环控制变量的值小于终限值。

5.2.3 使用循环语句时注意

1. 循环控制变量只能是实型变量，不能是整型变量或字符型变量，更不能是带有运算符的表达式。

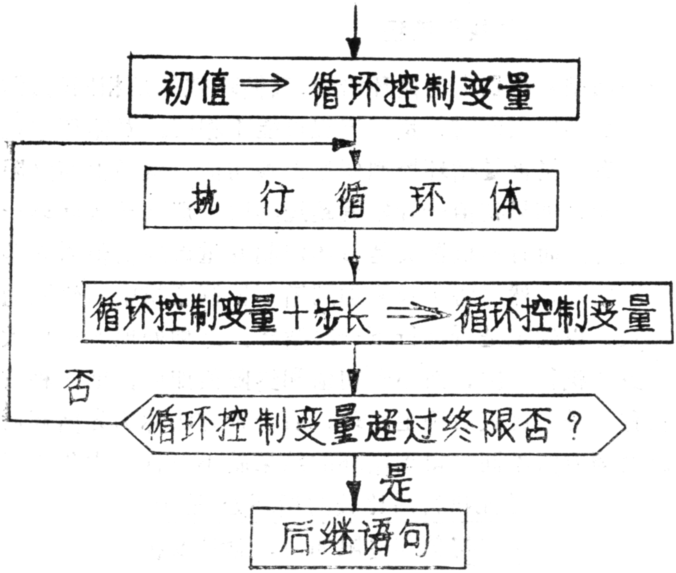


图5.1　循环语句处理示意图

例5.5　求1＋2＋3＋…＋100的值。

```
10  S＝0
20  FOR  I%＝1  TO  100
30  S＝S＋I%
40  NEXT  I%
50  PRINT  S
60  END
RUN ↵
? Syntax  error  in  20
```

第20句发生语法错。原因就在于循环控制变量I%是整型

变量。

2. 如果循环控制变量的初值一开始就超过终限值，那么循环也要执行一次。（注意“超过”是什么意思）

例5.6 我们用I来表示循环控制变量，用K来记录循环的次数，用S来代表累加和。

```
10 S=0：K=0
20 FOR I=100 TO 50 STEP 2
30 K=K+1
40 S=S+I
50 NEXT I
60 PRINT "K="；K，"S="；S，"I="；I
70 END
RUN ↵
K=1            S=100            I=102
```

从运行结果看，只循环了一次，因而K=1。累加和S=100，说明只把I的初值100加到S中去了。I=102说明把步长2累加到I中去了，因为初值为100，再累加上步长2，所以I=102。

3. 循环控制变量的值可以参与循环体中的运算，也可以只用于控制循环次数。但是切记：循环体中不能修改循环控制变量的值。如果修改了，就把正常的循环次序打乱了，使得我们难以控制。

例5.7 循环控制变量只用于控制循环次数。

```
10 W=0
20 INPUT A
30 FOR I=1 TO 100
40 W=W+A
```

```
50  NEXT  I
60 PRINT "W=" ; W, "A=" ; A, "I=" ;I
70  END
RUN ↲
? 4.72
W=472.000002          A=4.72          I=101
```

这里的I就只用于控制循环次数。如果在循环结束以后需要用到循环控制变量I的值，那么I的值就是最后那次循环的值（100）再加上步长（1），因而输出的I=101。这是因为中华学习机采用先执行后比较的缘故。

4. 初值、终限、步长均可以是复杂的算术表达式。

例5.8 初值、终限、步长是算术表达式的情况。

```
10  A=5 : W=0 : K=0
20  FOR  I=SQR(A)TO  EXP(A/3)/A
    STEP  A+A
25  K=K+1
30  W=W+I
40  NEXT  I
50  PRINT "K=" ; K, "W=" ; W
60  END
RUN ↲
K=1                  W=2.23606798
```

程序中K用于记录循环的次数，这里循环只执行了一次。

5. 步长可以是负值。

例5.9 步长为负值的情况。

```
10  S=0
20  FOR I=100  TO  1  STEP -1
30  S=S+I
40  NEXT  I
50  PRINT "S=" ; S
60  END
RUN ↵
S=5050
```

如果步长为负值（如本例中的-1），那么判断I是否超过终限1就用I<1来判断就行了。这个例子是把100，99，98，…，2，1累加起来。前面的求和是从小累加到大，这里是从大累加到小，实现方式不同，但效果完全一样。

6. 循环的次数可以用如下的公式来计算：

循环次数=INT（ABS(终限－初值)/步长）+1。

如果循环控制得不好，有可能丢掉一次应该执行的循环。

例5.10 下面的程序就少执行一次循环。

```
10  K=0
20  FOR  I=0.5  TO  5  STEP  0.3
30  K=K+1
40  PRINT "I=" ; I
50  NEXT  I
60  PRINT "循环次数=" ; K
70  END
RUN ↵
I=4.7
```

循环次数=15

以上输出的是最后那次循环的结果。由此可以看到，I=4.7时还应该再执行循环一次。但由于机内数据转换造成的误差，以致使本来应该循环的一次给丢掉了。当初值 步长为带小数时容易发生这种情况，避免丢掉循环的办法有二：

其一是在终限值上增加步长的一半。如上例终限取为5.15就不会丢失循环次数了。

例5.11 在终限值上增加步长的一半。

```
10  K=0
20  FOR  I=0.5  TO  5.15  STEP  0.3
30  K=K+1
40  PRINT "I=" ; I
50  NEXT  I
60  PRINT "循环次数=" ; K
70  END
RUN ↵
I= 5
循环次数=16
```

这就正确了。

其二是将循环参数扩大若干倍，使初值、终限，步长均成为整数。

例5.12 将初值、终限、步长扩大10倍使其成为整数。但循环中又缩小10倍后再使用之。

```
10  K= 0
20  FOR  I= 5  TO  50  STEP  3
30  K=K+1
```

```
40  PRINT "I="; I/10
50  NEXT I
60  PRINT "循环次数="; K
70  END
RUN ↲
I= 5
循环次数=16
```

显然，这个程序也是正确的。第40句中输出I值时，由于先前扩大了10倍，为了真实反映I值，所以需要缩小10倍。

7． 在循环体内的转向语句可以转到循环体外，但循环体外的转向语句不能转向循环体内。

例5.13　循环语句和转向语句的关系。

```
⋮
50 FOR I=1 TO 100
        ⋮
80      GOTO 110 ─┐
        ⋮         │
100 NEXT I        │
        ⋮         │
110 PRINT…     ←──┘
        ⋮
```

正确

```
⋮
50 FOR I=1 TO 100
        ⋮
80      W=W+I ←──┐
        ⋮         │
100 NEXT I        │
        ⋮         │
120 GOTO 80 ──────┘
        ⋮
```

错误

8． 在对话方式下使用循环语句时，FOR和NEXT必须写在同一行上（以是否敲↲作为换行标志，只要敲入↲，就表示换到新的一行。所以纸上书写的一行和机器内部的行是不完全相同的。这里所说的一行是指用↲隔开的行）且字

符个数（包括敲入的空格）不能超过239个。

9．FOR和NEXT形式的循环语句只能用于解决循环次数已知的那一类循环。如果循环次数未知或根据某种条件来确定是否继续循环，那么就只能用条件语句和转向语句配合才能完成。

例如：求$1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}+\cdots$，当$\frac{1}{n}<10^{-8}$时停止累加。这个问题不大好直接用FOR-NEXT循环语句来完成，因为当$\frac{1}{n}<10^{-8}$时需要循环多少次是无法事先知道的。这种情况只好用条件语句和转向语句来实现循环了。

5.3 多重循环

我们曾经提及，循环体中还可以包含循环语句，如果是这样就构成所谓的多重循环。

5.3.1 多重循环的结构

多重循环的结构如下：

```
   ┌FOR  i=i1  TO  i2  STEP  i3
   │             ⋮
外 │   内┌FOR  j=j1  TO  j2  STEP  j3
循 │   循│          〈循环体〉
环 │   环└NEXT  j
   │        ⋮
   └NEXT  i
```

这是一个二重循环，它在循环控制变量为I的循环中又套了一重循环变量为i的循环。通常把FOR j=…NEXT j称为内循环，而把FOR i=…NEXT i称为外循环。作为外循环的循环体，除了整个内循环以外，还可以包含其它一些语

句。

两重循环的执行情况如同一个大轮子带动一个小轮子，大轮（相当于外循环）转动一圈，小轮（相当于内循环）就要转动若干圈。

当然，还可以有三重、四重循环。CEC-BASIC至多允许嵌套10重循环。事实上，实际应用中很难有真正达到10重循环的情况。

例5.14 两重循环的例子。

```
10  W1=0：W2=0
20  FOR  A=1  TO  100
30  W1=W1+A
40  FOR B=2  TO 50 STEP  2
50  W2=W2+B*W1
55  PRINT "W2=" ; W2
60  NEXT  B
70  W3=W1/W2
80  PRINT "W3=" ; W3
90  NEXT  A
100 PRINT "程序结束"
110 END
RUN  ↵
W3=4.52488688E-05
程序结束
```

这是运行结束时输出的结果（输出了很多值，但我们仅写出了最后那个）。程序的执行过程是这样的：

①A赋给初值1后执行外循环的循环体。即30～80 句。

②第40句是内循环的开始，它将初值2赋给B，执行内循环的循环体。

③当执行到60句时，由于B的值没有超过终限值50，因而继续内循环，直到B超过终限后跳出内循环，执行其后继第70句。

④当执行到第90句时，由于A没有超过终限100，因而重复外循环。至此，外循环执行了一次，而内循环却执行了25次。重复以上过程，直到循环结束。

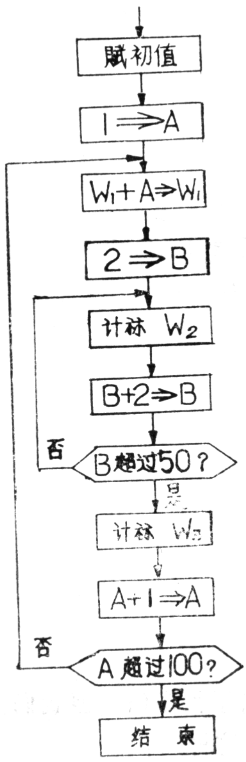


图5.2 两重循环框图

从上述说明可知，外循环A执行一次，内循环B就执行25次。整个内循环共要执行100×25＝2500次。为清楚起见请参见图5.2所示。

5.3.2 使用多重循环时注意

在使用循环语句编写循环程序时，除了前面关于循环语句的注意事项以外，对于多重循环的情况，我们还要强调以下几点。

1． 中华学习机CEC-I允许循环语句嵌套的最大重数是10重。通常情况下达不到10重。

2． 内外循环不能互相交叉。一个循环要么全部包含在另一个循环之中，要么全部在循环之外。请参见图5.3所示。

```
—FOR J=…
      ⋮
   —FOR I=…
        ⋮
   —NEXT I
      ⋮
—NEXT J
      ⋮
```

```
—FOR J=…
      ⋮
   —FOR I=…
        ⋮
   —NEXT I
      ⋮
   —FOR K=…
        ⋮
   —NEXT K
      ⋮
—NEXT J
```

（a）正确的循环嵌套

```
—FOR J=…
      ⋮
——————FOR I=…
           ⋮
—NEXT I,J
```

（b）　正确的公共出口

```
   —FOR J=…
        ⋮
——————FOR I=…
           ⋮
   —NEXT J
           ⋮
————————NEXT I
```

（c）错误的循环嵌套

图5.3　多重循环的嵌套结构

3. **多重循环嵌套可以共用一个出口NEXT，但必须由内到外逐层引出循环控制变量。**参见图5.3（b）所示。

例5.15 多个循环语句共用一个NEXT出口。

```
10  W=0
20  FOR  A=1  TO  50
30      FOR B=2  TO 60 STEP  5
40      FOR C=3  TO 30 STEP  3
50       W=(A+B+C)/(A*B*C)+W
60  NEXT  C, B, A
70  PRINT "W=" ; W
80  END
RUN ↵
W=147.237318
```

第60句是A、B、C三重循环的公共出回。在NEXT之后的循环控制变量必须按C，B，A（由内到外顺序书写且用"，"隔开。

4. **同一层循环最好在同一列上，这样书写，内外循环一目了然。**如例5.15所示。（由于版面原因，没有严格如此。）

5.4 循环程序实例

例5.16 假设a_1，a_2，…，a_{20}为已知值，求多项式$p(x)=a_1x+a_2x^2+a_3x^3+\cdots+a_{20}x^{20}$的值。

求多项式p（x）的值有两种方法实现。

①直接编写求a_ix^i累加和的程序。

```
5   PX=0
10  INPUT  X
```

```
20  FOR  I=1  TO  20
30  READ  A
40  PX=PX+A*X∧I
50  NEXT  I
60  PRINT "P(X)=" ; PX
70  DATA  1.32, 4.56, 7.88, 5.67,3.77, 6,2.8,
    3.11
75  DATA  7, 2, 3, 3.33, 5, 6, 1.66, 3.45,
    2.44, 7, 5.5, 5.678
80  END
RUN ↵
? 1.5 ↵
P(X)=50908.414
```

这种方法的最大优点在于简单和直观。但如果项数较多的话，运行速度可能很慢。因为每计算一项，X的方次都得重新计算。

②一个改进计算X方次的办法。

```
5   PX=0
10  INPUT  X
15  W=1
20  FOR  I=1  TO  20
30  READ  A
35  W=W*X
40  PX=PX+A*W
50  NEXT  I
60  PRINT "P(X)=" ; PX
70  DATA 1.32, 4.56, 7.88, 5.67, 3.77,6, 2.8,
```

```
    3.11
75  DATA  7, 2, 3, 3.33, 5, 6, 1.66, 3.45
    2.44, 7, 5.5, 5.678
80  END
RUN ↵
? 1.5 ↵
P(X)=50908.4138
```

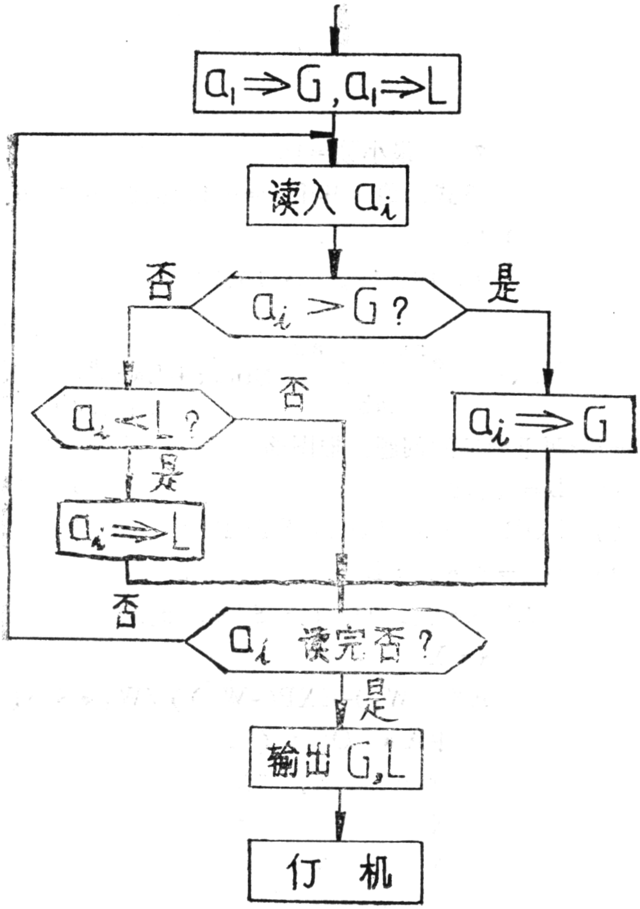
结果略有差异，主要是机器内部进行数据转换时造成的。这里仅增加了15句和35句。随着循环的执行，第35句就能形成X、X^2、X^3、…、X^{20}。虽然增加了两个语句，但机器运行时间提高不少，还是划算的。

例5.17　设有a_1，a_2…，a_{20}共20个数，找出其中的最大数和最小数。

设G代表最大数，L代表最小数。

解决问题的思路是：将第一个数分别送到G和L中。再读入下一个数，如果这个数比G大，那么说明该数有可能成为最大数。因此，将该数送到G中，如果这个数小于G，那么再把该数同最小数L比较。如果该数比L还小，那说明该数有可能是最小数。否则，说明该数既不可能是最大数也不可能是最小数，因而被淘汰。再读入下一个数，重复以上过程，最终一定能够找出最大数和最小数。框图如后。

```
10  READ  A
20  G=A: L=A
30  FOR  I=1  TO  19
40  READ  A
50  IF A>G THEN G=A: GOTO 70
60  IF A<L THEN L=A
```



70 NEXT I

80 PRINT "最大数＝"；G，"最小数＝"；L

90 DATA 12，34，23，56，45，78，123，567，
34，88

```
95  DATA  77, 78, 56, 90,  46, 76, 32, 10, 56。
    99
100 END
RUN ↲
```

最大数=567　　最小数=10

例5.18　求下式的值，其中X=0.1，0.3，0.5，0.7，…，1.5。

a=3，7，11，15，19。b=15.78。

$$Yax=\frac{e^{ax}-e^{-ax}}{2ab}Sinax+Lan\frac{a+x}{2ab}$$

这是一个两重循环的问题。框图略。

```
10  B=15.78
20  FOR A= 3  TO 19 STEP  4
30  W1= 2 * A * B
40  FOR X=0.1 TO 1.6 STEP 0.2
50  W2=a * X
60  Y= ( EXP(W2)-EXP(-W2) ) /W1 * SIN(W2)
          +A * LOG( ( A+X ) /W1 )
70  PRINT "A=" ; A, "X=" ; X, "Y=" ; Y
80  NEXT  X
90  NEXT  A
100 END
RUN ↲
A=19          X=1.5          Y=-889766765
```

我们只写出了运行结果的最后一组值。需要提及的是：在X的循环中，如果终限取值为1.5的话，由于数据转换误

差，造成X只能取到1.3,因而丢掉一次循环，所以我们把终限加大了半个步长成为1.6，因而避免了循环次数的丢失.请读者使用时特别注意。

例5.19 求一个三位的整数，其个位数字与十位数字之和能够被 2 整除；个位数字与百位数字之和能够被 3 整除；十位数字与百位数字之和能够被 4 整除。

我们用A、B、C分别代表所求整数的个位、十位和百位数字。而A、B、C分别可以取1～9中的某一个数字，如果这些数字能够满足所给条件，那么这时的A、B、C就是所求三位整数的个位、十位和百位。这个程序可用三重循环来实现。

```
10  FOR  A=1  TO  9
20  FOR  B=1  TO  9
30  FOR  C=1  TO  9
40  IF  A+B=INT((A+B)/2)*2  AND
        A+C=INT((A+C)/3)*3  AND
        B+C=INT((B+C)/4)*4  THEN 60
50  NEXT  C, B, A
55  PRINT "不存在满足条件的数"：GOTO  75
60  W=C*100+B*10+A
70  PRINT "这个三位数是"；W
75  PRINT "程序结束"
80  END
RUN ↵
这个三位数是531
程序结束
```

在该例中，如果给定的条件满足，就直接跳出循环打印

该数据。这种不通过NEXT出口而中途跳出循环的情况称为循环的非正常出口。通过逐层循环的NEXT而依次退出循环的情况称为循环的正常出口。在该例中如果是正常出口，则表示不存在满足条件的数，因而第55句给出了找不到这种数的标志。

问题与思考

1．用循环语句能够解决的问题用条件语句和转向语句也一定能够解决，反之亦然，这种说法正确吗？为什么？

2．为什么转向语句不能转入循环体？如果强行转入会发生什么情况？

3．转向语句经过FOR入口转入循环体行吗？为什么？

4．循环语句的终限值肯定比初值大吗？

5．循环控制变量的最后值一定是终限值吗？举例说明。

6．为什么最好把与本层循环无关的量提到循环之外？

习　题

1．指出下列程序的运行结果。

①
```
10 FOR I=3 TO 8 STEP 2
20 PRINT I
30 NEXT I
40 PRINT I
50 END
RUN ↵
```

②
```
10 FOR X=3 TO -3 STEP -3
20 PRINT X
```

```
30 NEXT X
40 END
RUN ↵
```

③
```
10 FOR A=1 TO 5 STEP 3
20 FOR B=A+1 TO 5 STEP 3
30 PRINT B
40 NEXT B
50 NEXT A
60 END
RUN ↵
```

④
```
10 FOR A=1 TO 5 STEP 2
15 PRINT "A=" ; A; " " ;
20 FOR B=A TO 5 STEP 2
25 PRINT "B=" ; B; " " ;
30 FOR C=B TO 5 STEP 2
35 PRINT "C=" ; C;
40 NEXT C
50 NEXT B
60 NEXT A
70 END
RUN ↵
```

⑤
```
10 FOR X=1 TO 3
20 FOR Y=1 TO 3
30 FOR Z=1 TO 3
40 IF X+Y+Z>5 THEN 95
50 IF X+Y+Z=5 THEN 85
60 IF X+Y+Z<>5 THEN 75
```

```
70  NEXT  Z
75  PRINT  X, Y, Z
80  NEXT  Y
85  PRINT  X, Y, Z
90  NEXT  X
95  PRINT  X, Y, Z
100 END
RUN  ↵
```

2. 指出程序的书写错误。

①
```
10  FOR  X$="3" TO "5"
20  PRINT  X$
30  NEXT  X$
40  END
```

②
```
10  FOR  X%=3  TO  5
20  PRINT  X%
30  NEXT  X%
40  END
```

③
```
10  FOR  X+Y=A  TO  B  STEP  A+C
20  PRINT  X+Y
30  NEXT  X+Y
40  END
```

④
```
10  FOR  X=1  TO  100
20  W=W+X
25  IF  W>100  GOTO  60
30  PRINT  W, X
40  NEXT  X
45  W1=W*W
```

```
50  GOTO  20
60  PRINT  W
70  IF  W>1000  THEN  30
80  END
```

⑤
```
10  FOR  A=1  TO  100
20  W=A*A*A
30  FOR  B=1  TO  5
40  W=W+B
50  PRINT  W
60  NEXT  B
70  IF  B>100  THEN  40
80  GOTO  20
90  PRINT  A
100 NEXT  A
110 END
```

⑥
```
10  FOR  A=1  TO  7
20  FOR  B=2  TO  10  STEP  3
30  W=A*B+A/B
40  PRINT  W
50  NEXT  A
60  PRINT  A
70  NEXT  B
80  END
```

3. 已知x_1y_1，x_2y_2，x_3y_3，…，$x_{20}y_{20}$的值，求

$W=\sqrt{x_i-y_i}$的值。当$x_i-y_i<0$时打印出错标记，继续进行下一次求值运算。

4． 求10000以内能够除3余2的数，并统计这种数的个数。

5． 编写0°～180°之间每隔1°的正弦、余弦、和弧度值表。并按如下格式输出：

三角函数表

角 度	弧 度	SIN	COS
0	…	…	…
1	…	…	…
2	…	…	…
⋮			
180	3.14159	…	…

6． 已知：C_1，C_2，…C_{10}和X的值，求

$$C(x)=\sum_{i=1}^{10}\left(C_i x \prod_{j=1}^{4} \cos jx\right)$$的值。

7． 学校把23本书分给获甲乙丙丁四等奖的四个班。要求甲等奖分得的书不多于9本，乙等奖不多于8本，丙等奖不多于7本，丁等奖不多于6本，并且每个班至少分得1本，还要求获奖越高，分得的书越多。问甲乙丙丁四等奖获得者各应分得多少本书？请编写出解决该问题的程序。

8． 找出数字符号互不相同的一个四位整数，使得前两位数字之和的平方等于后两位数字之和的立方，如果没有这样的四位数，则给出说明。

第六章　数组和维语句

6.1 什么是数组？

直到目前为止，程序中用到的所有变量，无论是数值型变量还是串型变量，它们均是以个体面目而出现的。这尤如一个普通公民，他（她）既不代表某个单位，也不代表某级组织。但是正如现实生活中一样，若干个体可以组成一个集体。例如：一个学生班级，一个学校，一个部门，一级组织等等都是一种典型的集体。在BASIC语言中，用于描述个体的是前面章节中介绍和使用的变量、常数和字符串。而用于描述若干个体组成的集体则是数组。简单说来，数组就是若干同类数据所组成的集体。集体中的单个成员，就叫数组的元素。全班学生的成绩数据可以组成一个数组，每个学生的成绩均是数组元素。而（5，10，25，30，40，80，100）和$\begin{pmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 13 & 15 & 17 \end{pmatrix}$都可以用数组来表示。数组中包含的数据个数可以很多，也可以很少。对于数组我们有如下的要求：

1. 数组的名字

对任何相同类型的数据集合，如果我们需要作为数组的话，都必须给该集合取一个名字，这就是数组名。数组名的构成方法同变量名的构成方法一样。

2． 分配存贮空间

任何一个数组，一经说明，系统就必须在内存中为之分配存贮空间。为了分配最适合的内存空间，因而说明数组时必须指出它所需要的最大存贮空间。我们把这个最大的存贮空间称为数组的大小。

3． 数组元素

定义数组是按一种集体的方式来说明的。但实际使用时又是按个体形式来引用的。对数组中的个体我们称为数组元素，数组元素也是一种变量，称为下标变量。数组元素的位置序号是重要的，因为对数组元素的存取就是通过指出元素在数组中的位置序号来实现的。这种存取方法，尤如教师只需说出座位号（某行某列）就可以指定学生一样。

6.2 数组的表示方法

程序中如何表示一个数组呢？这是用维语句来说明的。

6.2.1 维语句的格式

格式：DIM 数组名$_{-1}$(下标$_{-1}$，下标$_{-2}$，…)〔，数组名$_{-2}$(下标$_{-1}$，…)…〕

作用：定义一个或多个数组，并为数组在内存中分配存贮空间。

其中：DIM 是数组说明符。数组名的构成方法同变量名一样。下标指明数组元素能够分配的最大存贮空间，或者看成是数组元素能够变化的最大范围。下标的个数称为数组的维数。

1． 一维数组

能够用一个下标来描述的数组我们称为一维数组。例如：

(50，25，37，19，82，73，95)数据集，可以用一维数组来描述。假定该数据集的名字为A，共有7个元素，用维语句说明如下：

DIM A(6) 由于分配空间总量从第0号开始分配，因而A的下标为6。7个元素的位置分别是：A(0)，A(1)，A(2)，A(3)，A(4)，A(5)，A(6)。这里的A(0)，A(1)，A(2)，…A(6)就称为下标变量。

以上我们定义了一个一维数组A。如果我们将50送到A(0)，25送到A(1)，37送到A(2)，…，按这种顺序将(50，25，37，19，82，73，95)分别送到A(0)—A(6)中，那么A(4)的值就是82，A(6)的值是95。圆括号中的号码就是数组元素在数组中的位置。定义一个数组只是把存贮位置固定下来了，但该位置具体是什么值，还要看我们送的是什么数据。尤如座位定了，但某个座位是张三吗还是李四，那就看具体是哪个人坐在那里了。直观说来，凡是排成一队的数据，我们都可以用一维数组来刻划之。

2．二维数组

类似地，用两个下标来描述的数组我们称为二维数组。在实际生活中，凡是能够排成行和列(长方形、正方形)的的数据我们都可以用二维数组来描述。如：

$$B=\begin{pmatrix}11 & 12 & 13 & 14\\ 21 & 22 & 23 & 24\\ 31 & 32 & 33 & 34\end{pmatrix}$$ 用维语句描述如下：

DIM B(2，3)，数组B共有三行四列(包括第0行

第0列)。这里的B是数组名字，2和3都是下标。下标2表示数组B的行号变化范围是0到2，下标3表示数组的列号变化范围是0到3。

要表示数组B中的某一个元素，只能用下标变量的形式。如：B(0，1)表示数组B中第0行第1列的元素，B(1，3)表示数组B中第1行第3列的元素。其余类似。数组说明只确定了元素的位置，但该位置究竟存放的是什么值，这同数组获得数据有关。由于下标变量仍然是一种变量，因而根据需要，我们可以给下标变量赋一定的值。如B(1，2)=23，B(2，3)=34等。

按照上述方法，我们还可以定义三维数组、四维数组等等。CEC-BASIC至多可以定义到88维，但通常使用中一、二、三维数组最为普遍。

3. 下标和下标变量

在定义和引用数组元素时，下标和下标变量是两个不同而又有联系的概念。在维语句DIM中，下标用于刻划数组存贮空间的大小，而在下标变量中，下标却用于确定元素在数组中的位置。

下标变量是指数组中的一个元素。如：B(1，2)，B(0，3)就是下标变量。下标变量中表现出来的下标可以是一个常数，也可以是算术表达式。如：B(1，2)中的下标1，2是具体的数值常数。如：B(I+1，J)或B(I，J+1)中的I+1和J+1，均是算术表达式。

6.2.2 使用维语句时应注意

定义数组的方法是简单的。但实用中需要注意以下几点：

1. 数组一定要先说明后使用。

程序中用到的数组，必须在存取之前用DIM来定义。如程序中需用到W数组，可作如下形式的说明：

10 DIM W(10，10)　　这是一个11行11列的数组。如果一个数组最多11个元素(下标是10)，那么这种数组可以不用DIM说明而直接使用。这种情况称为数组的隐式说明，直接用维语句DIM来描述的数组称为显式说明。

2. 用一个DIM说明符可以定义多个数组名，但各个数组名都必须给出各自的下标。如果DIM定义了多个数组名，那么各个数组名之间必须用逗号“，”隔开。如：

10 DIM A(3，5)，B(10，8)，C(10，10，10)

这里A、B是二维数组，C是一个三维数组。A、B和B、C之间用“，”隔开。

3. 如果一个数组是多维数组，那么描述维数的下标之间也必须用“，”隔开。如：

10 DIM W(10，20，30)

这是一个三维数组。10，20，30之间均用“，”隔开各个不同的下标。

4. 下标允许的最大值是32767，最小值是0。数组的最大空间是(下标值＋1)相乘的积。如：

10 DIM A(10)，B(10，20)，C(5，6，7)

数组A的最大空间＝(10＋1)＝11

数组B的最大空间＝(10＋1)×(20＋1)＝231

数组C的最大空间＝(5＋1)×(6＋1)×(7＋1)＝336

5. 在引用下标变量时不能超过下标范围。如：

10 DIM W(10，15)

下标变量W(5，10)是正确的，而W(11，3)或W

(5，18)均是不正确的，因为下标超界了。即是说，在W中找不到第11行3列和第5行18列这两个元素(位置)。

6. 在同一程序中，一个数组名只能用DIM说明一次，但可以使用多次。

例6.1
```
10 DIM A(10, 10), B(15, 15)
……
50 DIM A(35, 30)
…
```

这里的数组A定义了两次，这是不正确的。

7. 下标变量中的下标可以是一个具体的数值常数，也可以是一个算术表达式，而维语句DIM中的下标只能是正整数。

例6.2
```
10 DIM A(10, 10)
…
40 I=3 : J=5
50 A(I+1, J+2)=153
…
```

这里的I+1和J+2就是算术表达式。

8. 在定义整数组和字符串数组组时，必须在数组名之后给出整数或字符串的标志。如：

```
10 DIM X$(10, 10), M%(5, 5)
```

这里的X$是字符串数组，其数组中的元素只能取串型值。M%是整数组，数组元素只能是整型数值。

6.2.3 如何给数组赋值

用DIM定义数组，只是给数组分配存贮空间，但如何把具体的数据传送到相应位置呢？一般用循环语句来实现。

1．一维数组如何得到数据

假设有一个一维数组A＝（10，30，50，15，80,28,），用程序来描述就是：

例6.3 向一维数组传送数据。

```
10 DIM A（5）
20 FOR I＝0 TO 5
30  READ A（I）
40 NEXT I
50  PRINT "A（4）＝"；A（4）
60 DATA 10，30，50，15，80，28
70 END
RUN
A（4）＝80
```

第10句定义一维数组A，它只有6个元素。第20～40句对数组A读入数据。一重循环结束后就把数据区提供的数据分别送到A的各个元素的位置上。第30句中的A（I）是一个下标变量，随着I值的不同（0～5），就提供了数组A的各个位置，因而能够把相应的数据送到数组的相应位置。第50句打印出下标变量A（4）的值，以验证所需数据是否传送到相应位置。

例6.4 生成数组B＝(10，20，30，40，50，60，70，80，90，100）

```
10  DIM B（9）
20  FOR I＝0 TO 9
30    B（I）＝I*10+10
40  NEXT I
50  PRINT "B（0）＝"；B（0），"B(8)＝"；
```

```
    B(8)
60  END
RUN↲
B(0)=10                    B(8)=90
```

由于数组元素的值是有规律的，所以就不必再用数据区的办法来提供数据了。随着循环语句的执行，第30句的B(I)=I*10+10就能自动生成10，20，30，…，100的数据并送到相应位置。

2. 二维数组如何得到数据

设有二维数组：

$$W=\begin{pmatrix} 35 & 72 & 48 & 63 & 94 \\ 101 & 32 & 49 & 53 & 68 \\ 11 & 75 & 77 & 69 & 61 \\ 54 & 47 & 98 & 78 & 87 \end{pmatrix}$$

显然这是一个四行五列的二维数组。

例6.5 按行将数据送到W的各个元素位置。

```
10  DIM W(3, 4)
20  FOR I=0 TO 3
30    FOR J=0 TO 4
40      READ W(I, J)
50    NEXT J
60  NEXT I
70   PRINT "W(3, 1)="; W(3, 1)
72  DATA  35, 72, 48, 63, 94
74  DATA  101, 32, 49, 53, 68
76  DATA  11, 75, 77, 69, 61
78  DATA  54, 47, 98, 78, 87
```

```
80  END
RUN↲
W(3,1)=47
```

上述程序中，把行的变化作为外循环（I），列的变化作为内循环（J），这种输入方式的实际效果就是按一行一行的顺序，依次把数据送到数组W中。当然我们也可以按一列一列的来输入数据。

例6.6 按列将数据送到W的各个位置。

```
10 DIM W(3,4)
20  FOR J=0 TO 4
30    FOR I=0 TO 3
40      READ W(I,J)
50    NEXT I
60  NEXT J
70  PRINT "W(3,1)=";W(3,1)
72  DATA 35,101,11,54
74  DATA 72,32,75,47
76  DATA 48,49,77,98
78  DATA 63,53,69,78
80  DATA 94,68,61,87
90  END
RUN↲
W(3,1)=47
```

按什么方式（按行或按列）输入数据，数据就应按相应的方式排列。最好是按行输入，这比较符合我们的阅读和书写习惯。总之，数据要传送到正确的数组位置。

例6.7 用程序来生成一个单位阵。

$$M=\begin{pmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{pmatrix}$$

这个数组中的元素值也是有规律的。除主对角线元素值为1以外，其余元素值均为0。

```
10  DIM  M(5, 5)
20  FOR  I=0  TO  5
30    FOR J=0  TO  5
40      IF I=J THEN M(I, J)=1:GOTO 55
50      M(I, J)=0
55      PRINT M(I, J); "  ";
60    NEXT J
65  PRINT
70  NEXT I
80  END
RUN↵
 1   0   0   0   0   0
 0   1   0   0   0   0
 0   0   1   0   0   0
 0   0   0   1   0   0
 0   0   0   0   1   0
 0   0   0   0   0   1
```

在程序中，随着二重循环的执行，40和50两句就能使主

对角元素值为1，非主对角元素值为0。

其它任何维数的数组均可按上述的原则来定义和生成。

6.3 数组的应用

在程序设计中，数组用得相当广泛。有些问题必须用数组才能解决。而有些问题可用数组也可以不用数组。

例6.8 假设实验中得到了d_1，d_2，…，d_{20}等20个数据，需要用这些数据来生成另外的20个数据，再找出生成数据中的最大值和最小值以及最大值、最小值所在的序号。

$$
W_i=\begin{cases}\dfrac{d_1+d_2}{2} & i=1\\[2ex] \dfrac{d_{i-1}+d_i+d_{i+1}}{3} & 1<i<20\\[2ex] \dfrac{d_{19}+d_{20}}{2} & i=20\end{cases}
$$

对于原始数据d_1，d_2，…，d_{20}，我们可以用一维数组D来表示。而生成的相应数据用一维数组W来表示。解决问题的思路是：

①定义和生成数组D，用D的数据来生成数组W。

②在数组W中找最大值和最小值，同时记录它们在W中的位置。

③用G代表最大值，K是最大值在数组W中的序号。用L代表最小值，J代表最小值的序号。

程序如下：

```
10 DIM D(20), W(20)
```

```
20 FOR I=1 TO 20
30  READ D(I)
40 NEXT I
50  W(1)=(D(1)+D(2))/2
55  W(20)=(D(19)+D(20))/2
60   FOR I=2 TO 19
70      W(I)=(D(I-1)+D(I)+D(I+1))/3
80  NEXT I
90  G=W(1):L=W(1):K=1:J=1
95  PRINT W(1); " ";
100 FOR I=2 TO 20
105  PRINT W(I); " ";
110  IF W(I)>G THEN G=W(I):K=I:
     GOTO 130
120  IF W(I)<L THEN L=W(I):J=I
130  NEXT I
135  PRINT
140  PRINT "最大值="; G, "序号是"; K
145  PRINT "最小值="; L, "序号是"; J
150  DATA 45, 67, 89, 12, 19, 50, 77,123,49,8
155  DATA 31, 32, 36, 38,33,41,44,51, 57, 58
160  END
RUN↙
```

（数组W中的数据略）

最大值=83.3333334　　　　序号是7

最小值=23.6666667　　　　序号是11

说明：

① D和W都分配了21个位置，而我们只用了1～20的位置，第0个位置没有用到。主要是使数组号码和实验数据脚标一致（都用1～20）。

② 各语句行的作用：

10句—定义D、W两个一维数组。

20～40句—读入数组D的原始数据。

50～80句—用D来生成数组W的各元素值。

90～130句—输出W的值，找出最大值、最小值以及它们各自的序号。输出W的值主要是为便于读者对照验证程序所求最大值、最小值是否正确。

140—145句—输出最大值、最小值及其序号。

150—155句—提供数组D的原始数据。

③ 运行时输出的W的值我们略去了没有写出来。

例6.9　用数组来统计学生成绩。

学号	语文	政治	数学	物理	化学	外语	生物	总分
6311	85	78	90	87	85	80	64	…
6314	75	67	85	78	77	63	75	…
………………								

假设全班共有10人，总分由计算机自动统计，我们只需按要求输入原始数据，统计结果由计算机输出。解题思路是：

① 用二维数组来描述学生成绩表。把学号和总分等计算在内共计9项（如上表）。

② 用循环语句把原始数据输入到数组，机器自动进行总分并将分数送到总分的相应位置。

③ 按表格形式输出数据。

程序清单如下：

```
10 DIM S(10, 9)
12 PRINT "学号"; TAB(5); "语文"; TAB
      (9); "政治"; TAB(13); "数学";
      TAB(17); "物理";
14 PRINT TAB(21); "化学"; TAB(25);
"外语"; TAB(29); "生物"; TAB (33);"总分"
16 PRINT
20 FOR I=1 TO 10
25   W=0
30   FOR J=1 TO 8
40      READ S(I, J)
50      IF J=1 THEN PRINT S(I, J); :
        GOTO 80
60      PRINT TAB(J*4-3); S(I, J);
70      W=W+S(I, J)
80   NEXT J
90   S(I, 9)=W
100   PRINT TAB(33); W
110  NEXT I
120  DATA 6311, 85, 78, 90, 87, 85, 80, 64
121  DATA 6314, 75, 67, 85, 78, 77, 63, 75
122  DATA 6316, 77, 75, 73, 80, 76, 74, 72
123  DATA 6318, 90, 93, 95, 92, 88, 89, 90
124  DATA 6322, 65, 64, 68, 75, 74, 77, 65
125  DATA 6324, 64, 63, 58, 68, 70, 72, 54
126  DATA 6330, 57, 58, 64, 65, 66, 63, 61
127  DATA 6335, 80, 78, 79, 77, 75, 74, 73
```

```
128  DATA 6336, 88, 89, 90, 87, 86, 83, 90
129  DATA 6342, 77, 74, 76, 78, 80, 71, 68
130  END
RUN↲     （运行结果如下）
```

学号	语文	政治	数学	物理	化学	外语	生物	总分
6311	85	78	90	87	85	80	64	569
6314	75	67	85	78	77	63	75	520
6316	77	75	73	80	76	74	72	527
6318	90	93	95	92	88	89	90	637
6322	65	64	68	75	74	77	65	488
6324	64	63	58	68	70	72	54	449
6330	57	58	64	65	66	63	61	434
6335	80	78	79	77	75	74	73	536
6336	88	89	90	87	86	83	90	613
6342	77	74	76	78	80	71	68	524

请读者自己分析程序的执行情况。

6.4 程序实例

例6.10 将下面两个二维数组对应元素相加。

$$\begin{vmatrix} 3 & 5 & 8 \\ 2 & 7 & 9 \\ 1 & 4 & 6 \end{vmatrix} + \begin{vmatrix} 4 & 6 & 3 \\ 9 & 2 & 8 \\ 4 & 6 & 3 \end{vmatrix} = \begin{vmatrix} 7 & 11 & 11 \\ 11 & 9 & 17 \\ 5 & 10 & 9 \end{vmatrix}$$

设二维数组分别为A、B、C。

```
10  DIM A(3, 3), B(3, 3), C(3, 3)
20  FOR I=1 TO 3
30  FOR J=1 TO 3
40  READ A(I, J), B(I, J)
50  C(I, J)=A(I, J)+B(I, J)
```

```
60  PRINT C(J, I); "  ";
70  NEXT J
80  PRINT
90  NEXT I
100 DATA 3, 4, 5, 6, 8, 3
102 DATA 2, 9, 7, 2, 9, 8
104  DATA 1, 4, 4, 6, 6, 3
110  END
RUN↲
7   11  11
11   9  17
 5  10   9
```

说明：为减少循环，本例中将A、B两个数组的值同时读入，然后相加送入C的相应位置。因而在DATA中排列数据时就只能将A、B数据交错排列。

例6.11　将下列矩阵进行转置(即行列元素交换位置)。

$$A=\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix} \quad \text{转置成} \begin{vmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{vmatrix}$$

```
10 DIM A(3, 3)
20  K=0
30  FOR I=1 TO 3
40    FOR J=1 TO 3
50    K=K+1
60    A(I, J)=K
70    PRINT A(I, J); "  ";
80    NEXT J
```

```
90  PRINT
100 NEXT I
110 PRINT "进行转置"
120 FOR I=1 TO 2
130   FOR J=I+1 TO 3
140   W=A(I, J)
150   A(I, J)=A(J, I)
160   A(J, I)=W
170   NEXT J
180  NEXT I
190   PRINT "打印转置阵"
200  FOR I=1 TO 3
210   FOR J=1 TO 3
220   PRINT A(I, J); "  ";
230   NEXT J
240   PRINT
250  NEXT I
260  PRINT "程序结束"
270  END
RUN↵（以下是运行结果）
 1      2      3
 4      5      6
 7      8      9
进行转置
打印转置阵
 1      4      7
 2      5      8
 3      6      9
```

程序结束

说明：

20～100句生成矩阵A。

120～180句用于将矩阵A进行转置。转置时需要将行列元素交换位置，因而引入一个中间工作变量W，其作用好比将两个杯子的水互换而需要第三个杯子一样。

200～250句打印出转置阵的各元素值。

例6.12　将n个元素按从小到大的顺序排列。

这里必须要用到数组，解决思路是：

①定义和生成一维数组A。

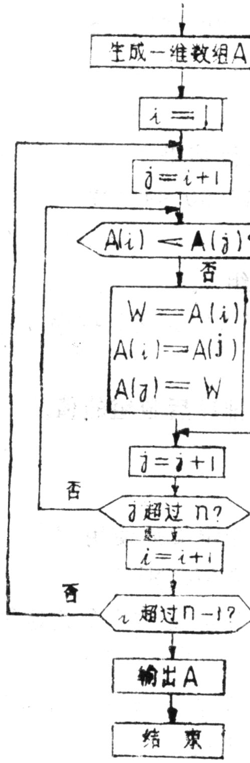
②通过比较和位置交换，把最小的数据移到数组的最前面，较大的数据移到数组的后面。

③通过n次比较可以得出一个最小值，再经过n-1次比较又可得到一个次小值。如此进行，直到把所有数据按从小到大排列为止。

④尽管处理的是一维数组，但必须用到二重循环。外循环控制将最小值移到数组的前面，其循环一次只能确定一个较小值。内循环控制使两个数据进行比较。内循环结束后才能确定一个值是否是较小。

先看后面的框图，然后根据框图再来读程序。

```
10  DIM A(10)
20  FOR I=1 TO 10
30    READ A(I)
40  NEXT I
50  FOR I=1 TO 9
60    FOR J=I+1 TO 10
70      IF A(I)<A(J) THEN 110
```




```
80          W=A(I)
90          A(I)=A(J)
100         A(J)=W
110   NEXT J
120   NEXT I
130   FOR I=1 TO 10
140    PRINT A(I); "  ";
```

```
150  NEXT I
160  DATA 60, 50, 80, 23, 75, 94, 19, 132,48,
   46
170  END
RUN↲          （运行结果）
19 23 46 48 50 60 75 80 94 132
```

说明：

20～40句用于生成数组A。

50～120句用于将数组A的元素值排序。I循环一次，就确定出一个较小值。

130～150句用于输出排序后数组的值。

问题与思考

1. 为什么数组必须先定义后使用？
2. 下标、下标变量有何区别？
3. 数组的维数就是下标变量的个数这种说法对吗？

习　　题

1. 指出下列程序的错误。

①
```
10 DIM A(10, 10)
20 FOR I=1 TO 10
30    FOR J=1 TO 10
40     DIM A(15, 15)
50     READ A(I, J)
60   NEXT J
70 NEXT I
80 DATA…（数据省略）
```

```
90 END
```

②

```
10  DIM A$(10, 10)
20  FOR I=1 TO 10
30  FOR J=1 TO 10
40    READ A$(I, J)
50    NEXT J
60  NEXT I
70  DATA 10, 20, 30, 80, 90, 100, 75,85,
    95, …
80  END
```

2. 指出下列程序运行的结果。

```
10  DIM  B(5, 5)
20  FOR I=1 TO 5
30    FOR J=1 TO 5
40    IF I-J>=0 THEN A(I, J)=1:GOTO
        60
50  A(I, J)=I-J+1
60  PRINT A(I, J); "  ";
70  NEXT J
80  PRINT
90  NEXT I
100  END
RUN↵
```

3. 用程序生成如下形式的数组。

① $\begin{vmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{vmatrix}$ ② $\begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{vmatrix}$

4. 将下列两个数组的对应元素进行交换。

$$A=\begin{vmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 13 & 15 & 17 \end{vmatrix}, \quad B=\begin{vmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{vmatrix}$$

5. 将下列两个数组相乘。

$$A=\begin{vmatrix} 15 & 24 & 31 \\ 52 & 28 & 19 \\ 11 & 14 & 17 \end{vmatrix}, \quad B=\begin{vmatrix} 24 & 28 & 29 \\ 31 & 42 & 15 \\ 7 & 9 & 2 \end{vmatrix}$$

第七章 BASIC函数

根据函数的作用和运算结果，我们把BASIC函数分成如下几类：算术运算函数、字符串处理函数、数据类型转换函数、格式打印控制函数以及自定义函数。

7.1 算术运算函数

算术运算函数可以参与BASIC算术表达式中的算术运算或单独进行运算以得出一个数值。

7.1.1 三角函数

1. 求正弦值

格式：SIN（X）

作用：求X弧度的正弦值。

例7.1 ？SIN（5）↵

　　　　－.958924274

注意：①变量X可以是任何合法的算术表达式。

②变量X必须用弧度来表示。

2. 求余弦值

格式：COS（X）

作用：求X弧度的余弦值。

例7.2 ？COS（5）↵

　　　　.283662186

3. 求正切值

格式：TAN (X)

作用：求X弧度的正切值。

例7.3　?　TAN (5)↵

　　-3.380515

4．求反正切值

格式：ATN (X)

作用：求X的反正切值。

例7.4　?　ATN (5)↵

　　1.37340077

注意：变量X必须在$-\frac{\pi}{2}\sim+\frac{\pi}{2}$之间。

对于三角函数，我们要强调以下两点：

①提供的实在参数（调用时的参数）必须是弧度，如果实际应用中是角度，那么必须把角度转换成弧度。按$180^\circ=\pi$，$1^\circ=\frac{\pi}{180}$公式进行转换。最好由机器自动进行转换。

②实在参数可以是数值常数、数值变量或任何合法的算术表达式。

7.1.2　其它数学函数

1．求绝对值

格式：ABS (X)

作用：求X的绝对值。

例7.5　?　ABS (-3)↵

　　3

2．求平方根

格式：SQR (X)

作用：求X的平方根。

例7.6 ? SQR(100)↙

10

注意：X的值只能是正数。

3. 求e^x的值

格式：EXP(X)

作用：求e^x的值。

例7.7 ? EXP(3)↙

20.855369

4. 求自然对数值

格式：LOG(X)

作用：求X的自然对数Lg(X)的值。

例7.8 ? LOG(3)↙

1.09861229

注意：X的值必须大于0。

5. 取整数

格式：INT(X)

作用：取最接近X而又不大于X的最大整数。

例7.9 ? INT(3.14159)↙

3

? INT(5.99998)↙

5

? INT(-3.14159)↙

-4

6. 求符号值

格式：SGN(X)

作用：取x的符号值，取值原则是：

$$若SGN(X)=\begin{cases}1，则X>0\\0，则X=0\\-1，则X<0\end{cases}$$

例7.10

```
? SGN ( 5 ) ↵
1
? SGN ( 0 ) ↵
0
? SGN ( -5 ) ↵
-1
```

7. 求随机数

格式：RND (X)

作用：根据X的值来产生一个随机数。原则是：

$$若RND(X)\begin{cases}X>0;产生0\sim0.999999999之间的一个随机数\\X=0，产生与前一个随机数相同的随机数\\X<0，产生与X相对应的一个随机数\end{cases}$$

例7.11 产生一系列随机数。

```
10 PRINT "RND(1)=" ; RND(1)
20 PRINT "RND(1)=" ; RND(1)
25 PRINT "RND(5)=" ; RND(1)
30 PRINT "RND(0)=" ; RND(0)
40 PRINT "RND(0)=" ; RND(0)
50 PRINT "RND(-5)=" ; RND(-5)
60 PRINT "RND(-3)=" ; RND(-3)
70 PRINT "RND(-5)=" ; RND(-5)
80 PRINT "RND(-3)=" ; RND(-3)
90 END
```

```
RUN↵
RND( 1 )=.273385388
RND( 1 )=.621966945
RND( 5 )=.934074496
RND( 0 )=.934074496
RND( 0 )=.934074496
RND(-5)=3.73720468E-08
RND(-3)=4.48217179E-08
RND(-5)=3.73720468E-08
RND(-3)=4.48217179E-08
```

综合考察以上算术运算函数，它们有如下一些共同点

①提供的参数可以是算术表达式。

②运算的结果均是数值。

②三角函数要求参数均用弧度。

在使用这些函数时还要注意：

①正确提供实在参数。例如有些参数要求是正数值，有些参数有范围限制，如求反正切值arctgX，要求 X 的 值在 $-\frac{\pi}{2} \sim +\frac{\pi}{2}$

②BASIC函数同数学函数在写法上的区别。

③函数是表达式而不是语句，因为它只能进行函数运算而不能提供操作动作。它可以是语句的组成部份。

7·2 字符串处理函数

为进行字符串处理，BASIC语言中提供了如下的字符串函数。

1.取左子字符串

格式：LEFT $(A $, n)

作用：取字符串A $中左边的几个字符。

例7.12　取出A $中左边的4个字符。

```
10  A$= "ABCDEFGH"
20  B$=LEFT$(A$, 4)
30  PRINT "B$=" ; B$
40  END
RUN
B$=ABCD
```

2.　取右子字符串

格式：RIGHT $(A $, n)

作用：取出字符串A $中右边的n个字符。

例7.13　取出A $中右边的4个字符。

```
10  A$= "ABCDEFGH"
20  B$=RIGHT$(A$, 4)
30  PRINT "B$=" ; B$
40  END
RUN↲
B$=EFGH
```

3.　取中间子字符串

格式：MID $(A $, n, m)

作用：取出字符串A $中从第n个字符开始的m个字符。

其中：n表示开始字符位置，m表示字符的个数。

例7.14　取出A $中间的字符。

```
10  A$= "ABCDEFGH"
20  B$=MID$(A$, 3, 4)
```

```
30  PRINT "B=$" ; B$
40  END
RUN↲
B$=CDEF
```

对于MID$函数，需要注意以下几点：

①如果m被省略，就表示取从第n个字符开始直到最后那个字符之间的所有字符。

例7.15　省写m的情况。

```
10  A$= "ABCDEFGH"
20  B$=MID$(A$, 3 )
30  PRINT "B$=" ; B$
40  END
RUN↘
B$=CDEFGH
```

②若n超过实际字符个数，则产生一个空串（屏幕上没有任何表示）。

例7.16　n大于A$中字符个数。

```
10  A$= "ABCDEFGH"
B$=MID$(A$, 10,10)
30  PRINT "B$=" ; B$
40  END
RUN↘
B$=            （这是空串）
```

③若m大于实际字符的个数，则有多少字符就取多少字符。

例7.17　m大于实际字符个数。

```
10  A$= "ABCDEFGH"
```

```
20  B$=MID$(A$, 5, 15)
30  PRINT "B$=" ; B$
40  END
RUN↵
B$=EFGH
```

④1≤n≤255，1≤m≤255，若m，n值超界，则出现“ILLEGAL QUANTITY ERROR”，表示不合法的参数出错。

4. 求字符串的长度

格式：LEN(A$)

作用：检测字符串A$中有多少个字符。

例7.18 求A$的字符串长度。

```
10  A$= "ABCDEFGH"
20  B=LEN(A$)
30  PRINT "B=" ; B
40  END
RUN↵
 8
```

对于字符串处理函数必须注意：

①函数中用到的A$可以是一个字符串常数、或字符串变量，也可以是字符串表达式。

②n和m也可以是算术表达式。

③字符串长度函数得到的是一个算术值。

7.3 数据类型转换函数

BASIC提供了将数值型转换成字符型以及把字符型

（仅由数字开头组成的字符串）转换成数值型的数据类型转换函数。

1. 将数值型转换成字符型

格式：STR$（〈算术表达式〉）

作用：将给定的算术表达式的值由数值型转换成字符型。

例7.19 将数值型转换成字符型。

```
? STR$ (10*10) ↵
100       (这是字符100)
10  A=10*10
20  B$=STR$(A)
30  PRINT "B$" =; B$
40  END
RUN↵
B$=100
? STR$ (3.72*10^15) ↵
3.72E+15      (字符串)
```

数值型转换成字符型以后就可参与字符串运算了。

例7.20 将字符串进行连接。

```
10  B$=STR$(3.72*10^15)+"+"+STR$(10*10)
20  PRINT B$
30  END
RUN
3.75E+15+100      (这些均是字符串)
```

2. 将字符型转换成数值型

格式：VAL（〈串表达式〉）

作用：将由数字开头的字符串转换成数值。

例7.21 将"123+ABC"转换成数值。

```
? VAL ( "123+ABC" ) ↲
123      (数值)
? VAL ( "123456789" ) ↲
123456789      (数值)
? VAL ( "A123+B456" ) ↲
0
```

对于VAL函数，并非所有字符串都能转换成数值。对于全部由数字符号组成的字符串，VAL函数可以将其完整地转换成数值。对于仅由数字开头的字符串，VAL只能将开头连续的数字符号转换成数值。对于非数字符号开头所组成的字符串，VAL只能转换成数值0。

3. 求数值对应的ASCII码字符。

格式：CHR$(〈算术表达式〉)

作用：给出算术表达式值所对应的ASCII码字符。

例7.22 求0～255之间的ASCII码字符。

```
10  FOR I=0  TO 255
20  PRINT CHR$(I); " ";
25  IF INT(I/10)*10=I THEN PRINT
30  NEXT I
40  END
RUN↲
```

屏幕上显示出所有字符（这里省略了）。第25句的作用是使每10个符号占一行，便于浏览。

注意：

①算术表达式的值必须在0～255之间。

②若算术表达式的值为实数，机器自动转换成整数后再进行函数运算。

③用该函数可以得到键盘上没有的许多符号。

4. 求字符的ASCII码

格式：ASC（〈字符串表达式〉）

作用：返回字符串中第一个字符的ASCII码值。

例7.23 求A的ASCII码值。

?ASC（"ABCD"）↵

65 （字母A的ASCII码值是65）

?ASC（"WXYZ"）↵

87 （字母W的ASCII码值是87）

注意：该函数每次只能返回字符串表达式中最左边那个字符的ASCII码值，而不能给出串中全部字符的ASCII码值。

7·4 控制输出格式的函数和命令

前面我们介绍和使用了PRINT语句，其一般格式应该是：PRINT〔打印格式〕〔表达式〕，其中的打印格式是表达式之后紧跟分号或逗号，这种情况我们已经介绍了。下面再介绍几个与之有关的函数。

7.4.1 格式及其作用

1. 空格函数

格式：SPC（〈算术表达式〉）

作用：以上一次输出的最后那个字符为基准，跳过若干个空格。算术表达式的值就是应该跳过的空格的个数。

例7.24 在两个输出数据间留出适当空格。

```
10  PRINT 125; SPC(3); 6789
20 END
```

```
RUN↙
125       6789
```

在数字125的5之后留出3个空格。即是说125和6789之间有3个空格。

注意：

①SPC函数只能放在PRINT之后用于跳过若干空格。

②算术表达式的值指出了空格的个数，其值必须在0～间255之间。

2. 将光标移到指定的行上

格式：VTAB(〈算术表达式〉)

作用：将光标移到算术表达式的值所指定的行上。

例7.25　将光标移到第3行上。

```
10  PRINT 125;
20 VTAB(3)
30  END
RUN
125
```

输出数值125以后，光标移到屏幕的第三行上。

注意：

①VTAB命令只能将光标上下移动而不能左右移动。如果光标当前所处的行号比VTAB中给出的行号小，那么光标向下移。如果当前行号比VTAB中指定的行号大，那么光标上移。

②在西文状态下，算术表达式的值只能在1～24之间(屏幕上所能显示字符的行号范围)。中文状态下只能在1～10之间。

3. 将光标移动若干列

格式：HTAB(〈算术表达式〉)

作用：将光标移动指定的列数。

例7.26 将光标右移10列。

```
10  PRINT 125;
20  HTAB(10)
30  PRINT 789
40  END
RUN
125     789
```

数值125和789之间空出了十个字符位置，这就是HTAB(10)的作用。

注意：

①算术表达式的值给出了左右移动的列数。

②在西文状态下，1～40为当前行（即光标所在的行），41～80列为下一行；在中文状态下，1～34列为当前行，35～68列为下一行。

③算术表达式的值只能在0～255之间。

4. 将光标定位在指定的列上

格式：TAB(〈算术表达式〉)

作用：以当前行上的第一列为基准，将光标定位在指定的列上。

例7.27 在指定位置上开始打印数据。

```
10  PRINT 125; TAB(10); 6789;
20  PRINT TAB(20); "AAAAA"
30  END
RUN↲
125          6789          AAAAA
```

注意：

①光标只能向右移而不能回退。即是说，如 果 先 用了TAB（20），就不能再在同一行上用TAB（15）。

②TAB只能在PRINT中使用。

③算术表达式的值只能在0～255之间。

5．求光标所处的列位置

格式：POS（X）

作用：返回光标所在的列号。如：

```
? POS（ 5 ）↵
3   （表示光标现在正处于第3列上）
```

注意：这里的X是无所谓的，实际上不起任何控制作用，因而该函数可以写成POS（ 0 ）。写其它值效果也一样。

7.4.2 打印格式的控制

在前面介绍的格式函数中，用得最多的是TAB函数。它可以使我们方便地把需要打印的字符安排在适当的列位置上。我们用几个例子来说明如何使用TAB函数。

例7.28 打印如图7.1所示的图形。

```
10  FOR I=1 TO 8
20  FOR J=1 TO I
30  PRINT TAB(15+J);"A";
40  NEXT J
50  PRINT
60  NEXT I
70  END
RUN↵
```

```
A
AA
AAA
AAAA
AAAAA
AAAAAA
AAAAAAA
AAAAAAAA
```

图7.1

运行结果如图7.1所示。

例7.29 打印图7.2所示图形。

```
10  FOR I=1 TO 8
20  FOR J=1 TO I
30  PRINT TAB(21+J-I); "A";
40  NEXT J
50  PRINT
60  NEXT I
70  END
RUN↵
```

运行结果如图7.2所示

```
       A
      AA
     AAA
    AAAA
   AAAAA
  AAAAAA
 AAAAAAA
AAAAAAAA
```

图7.2

例7.30 打印如图7.3所示图形。

```
10  FOR  I= TO 8
20  FOR J=1 TO 2*I
30  PRINT TAB(15-I+J); "A";
40  NEXT J
50  PRINT
60  NEXT I
70  END
RUN↵
```

```
       AA
      AAAA
     AAAAAA
    AAAAAAAA
   AAAAAAAAAA
  AAAAAAAAAAAA
 AAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
```

图7.3

运行结果如图7.3所示。

例7.31　打印如图7.4所示图形。

```
10  FOR I=1 TO 8
20  FOR J=1 TO 8
30  PRINT TAB(5+I+J); "A";
40  NEXT J
50  PRINT
60  NEXT I
70  END
RUN↵
```

运行结果如图7.4所示。

```
 AAAAAAAA
  AAAAAAAA
   AAAAAAAA
    AAAAAAAA
     AAAAAAAA
      AAAAAAAA
       AAAAAAAA
        AAAAAAAA
```

图7.4

上面几个例子中用到了 TAB 函数，其作用均是控制字符打印的位置。要使字符打印在所希望的列位置上，关键是设法使TAB(〈算术表达式〉)中算术表达式的值能够恰到好处。只要找出了打印字符的排列规律，算术表达式就不难构造出来了。

7.5　自定义函数

以上介绍的这些函数是由系统提供给用户直接使用的，通常称为标准函数。这些函数功能专一，调用简便。为使用

户能够象使用标准函数那样方便，BASIC又提供了用户定义自己所需函数的能力。这就是所谓的自定义函数。

格式：DEF FN〈自定义函数名〉(〈形式参数〉)
=〈算术表达式〉

作用：允许用户定义自己所需的函数。

其中：DEF FN是自定义函数的标志。自定义函数名的构成法同变量名相同。形式参数是一个实型算术变量，算术表达式是需要计算的式子。

我们用例子来说明如何定义和使用自定义函数。

例7.32 求下式的值。

$$W=\frac{b-a}{4}\left(\frac{\sqrt{1+a^2}}{2}+2\sqrt{1+(a+h)^2}+\frac{\sqrt{1+(a+2h)^2}}{2}\right)$$

其中：$a=5$，$b=8$，$h=\frac{b-a}{2}$

我们用自定义函数来解决该问题。

```
10  DEF FN F(X)=SQR(1+X*X)
20  A=5:B=8:H=(B-A)/2
30  W=(B-A)/4*(FNF(A)/2+2*FNF(A+H)
    +FNF(A+2*H)/2)
40  PRINT "W="; W
50  END
RUN↵
W=14.8001888
```

说明：

10句定义了一个自定义函数F(x)，它用于计算F(x)=

$\sqrt{1+x^2}$ 式子的值。

30句中调用了10句中定义的函数F(X)。函数F(X)如同标准函数的使用一样方便。

使用自定义函数时必须注意：

1． 自定义函数必须是数值型函数，不允许是字符串型。

2． 自定义函数必须先定义后使用。

3． 自定义函数中至多只能有一个形式参数，该形式参数在算术表达式中可以出现多次，也可以一次都不出现。

例7.33 算术表达式中不带参数。

```
10  DEF FN P(X)=SQR(100)+LOG(5)
20  PRINT "P(X)=" ; FNP(0)
30  END
RUN↲
P(x)=11.6094379
```

算术表达式中没有一个变量。P(X)中的X称为虚变量，它只是在形式上占据一个位置，实际上并不起任何作用。

4． 同一程序中可以对同一自定义函数进行重新定义。

例7.34 对自定义函数的重新定义。

```
10  DEF FN F(X)=SQR(1+X*X)
20  W1=FNF(5)+FNF(8)
30  DEF FN F(Y)=(LOG(Y)+EXP(Y))∧2
40  W2=FNF(3)
50  PRINT "W1=" ; W1, "W2=" ; W2
60  END
RUN↲
W1=13.1612773     W2=448.768179
```

在上述程序中，第10句定义了自定义函数F(X)，但在第30句中又对自定义函数进行了重新定义。这是允许的。

5. 在调用自定义函数时，必须写成FN〈变量名〉(〈实在参数〉)的形式。如上例中的FNF(5)，FNP(0)等。自定义函数是一种算术表达式，因而，凡是算术表达式能够出现的地方，自定义函数均可出现。

问题与思考

1. 自定义函数与标准函数有何区别？

2. 下面形式的数组应如何定义和使用？请用程序来实现。

姓名	学号	语文	数学	外语	总分
张三	612	85	90	86	…
李四	614	78	79	80	…
王五	620	81	83	78	…

习　　题

1. 用自定义函数编写程序。

$$M=\frac{1+\sin\alpha+\sin^2\alpha}{1+\cos\alpha}\left(\sqrt{\frac{1+\sin\beta+\sin^2\beta}{1+\cos\beta}}-\frac{1+\sin\gamma+\sin^2\gamma}{1+\cos\gamma}\right)$$

其中：$\alpha=30°$，$\beta=38°$，$\gamma=58°$

2. 编写打印九九表的程序。排列形式为：

1 * 1=1

2 * 1=2 2 * 2=4

3 * 1=3 3 * 2=6 3 * 3=9

4 * 1=4 4 * 2=8 4 * 3=12 4 * 4=16

5 * 1=5 5 * 2=10 5 * 3=15 5 * 4=20 5 * 5=25

3. 编写打印下列学生成绩统计表的程序。

学生成绩统计表

学号	姓名	语文	政治	数学	物理	外语	化学	总分
613	张三	85	78	82	73	77	70	…
624	李四	92	91	88	85	85	80	…
⋮	…							

第八章　子　程　序

重复执行某段程序，在大多数情况下利用循环语句来实现是方便的。但在某些情况下用子程序技术却更为简便。

8.1　转子语句和返回语句

在实际的程序设计中，我们经常都会碰到这样的问题：一个大型问题中包含了若干相对独立的小问题。对于这种情况，程序设计的一般经验是将大问题用一个总体程序来实现，称为主程序。每一个相对独立的小问题各自单独对应一个程序，称为子程序。当主程序中需要解决某个小问题时，就调用它所对应的子程序。通常，主程序向子程序提供数据，子程序向主程序返回运算结果。

8.1.1　**语句格式及其作用**

格式$_1$：GOSUB 〈行号〉

作用：无条件转向〈行号〉所代表的子程序去执行。当遇到子程序中的返回语句时再把控制转到该转子语句的后继语句。

其中：GOSUB是转子语句的标志。行号是子程序的入口语句行号。

格式2：ON n GOSUB 行号$_{-1}$〔,行号$_{-2}$…〕

作用：根据n的值转到相应行号所代表的子程序去执行。

其中：ON n是转子条件，n是一个算术表达式且其值是整数（$0 \leqslant n \leqslant 255$）。如果n为实数，机器将自动转换为整数。如果$n=0$或n大于子程序的最大行号，那么就执行转子语句的后继语句。n的值应该同指出的子程序入口语句行号的顺序号相对应。$n=1$，转到行号$_1$，$n=2$，转到行号$_2$，其余类推。

格式：RETURN

作用：从子程序返回到相应转子语句的后继语句去执行。

RETURN这几个字母是返回语句的标志。它表示依次敲入R、E、T、U、R、N这几个字母，而不是代表回车键。一个子程序中至少有一个返回语句。

如何利用转子语句和返回语句来编制程序呢？我们看下面的例子。

例8.1 用主程序和子程序的形式来编写程序。

$$W=\frac{b-a}{4}\left(\frac{\sqrt{1+a^2}-\sqrt[3]{1-a^2}}{2}+4\,\frac{\sqrt{1+(a+h)^2}-\sqrt[3]{1-(a+h)^2}}{2}+\frac{\sqrt{1+(a+2h)^2}-\sqrt[3]{1-(a+2h)^2}}{2}\right)$$

其中：$a=5$，$b=8$，$h=\dfrac{b-a}{2}$

在上式中，$\dfrac{\sqrt{1+x^2}-\sqrt[3]{1-x^2}}{2}$形式的式子出现了三

次，我们就把该式作为子程序。

主程序：

```
10   A=5:B=8:H=(B-A)/2
20   X=A
30   GOSUB 1000
40   F1=F
50   X=A+H
60   GOSUB 1000
70   F2=F
80   X=A+2*H
90   GOSUB 1000
100  F3=F
110  W=(B-A)/4*(F1+4*F2+F3)
120  PRINT "W=" ; W
130  END
```

子程序：

```
1000 S=X*X
1010 F=(SQR(1+S)-(1-S)∧(1/3))/2
1020 RETURN
RUN↵
W=14.8001888
```

说明：

10～130句是主程序段。它的作用是向子程序提供数据，保存子程序中返回的运算结果。

20,50,80句均是向子程序提供数据。

30,60,90句均是调用子程序。子程序入口语句行号是1000。

40,70,100句是保存子程序中返回的计算结果。

110句是利用保存的结果计算W的值。

130句是主程序的结束标志。在带有子程序的程序设计中，主程序一定要有END语句（命令）。

1000～1020句是子程序。显然，这里的子程序相当简单。子程序中至少有一个RETURN语句。

主程序和子程序的执行情况如下：

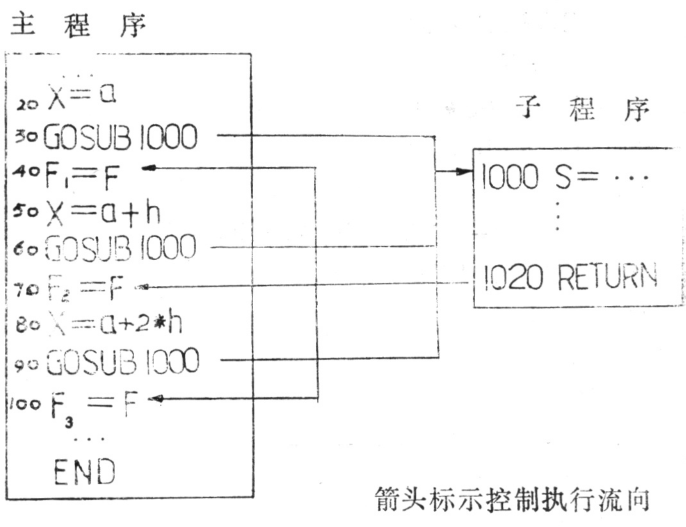


图8 1 主-子程序执行流程示意

20句向子程序中的变量X提供数据，30句转到行号为1000的子句去执行，当执行到子程序的RETURN语句时，控制又返回到主程序中相应转子语句（30句）的后继语句（40句）去执行。第40句把子程序中计算的结果F保存到中间变量F_1中。第50句向X提供数据A+H。第60句又转到子程序去执行。重复以上过程，直到调用完毕。

通过分析以上主—子程序的执行情况，我们可以得出主程序和子程序调用的一般方式是：主程序向子程序提供数据——转向子程序——执行子程序——返回主程序——子程序向主程序提交运行结果——再执行主程序。

例8.2　求$ax^2+bx+c=0$的根。其中a，b，c可以取不同的几组数据。为便于比较，请将方程的根列成表格形式，如下所示：

A	B	C	X_1	X_2	备注
3	4	5	…	…	…
8	20	10	…	…	…
……					

备注项用于说明方程得到的是实根吗还是虚根（如果是虚根，那么表中X_1是实部，X_2代表虚部）。

我们把提供数据（a，b，c的值）作为主程序，把求解方程的根作为子程序$_{-1}$（入口是1000）。把格式打印作为子程序$_{-2}$（入口是5000）。请看下面的程序。

主程序：

```
10  PRINT "————————————————————————"
15  PRINT "|"; TAB(3); "A"; TAB(6); "B";
    TAB(9); "C";
20  PRINT TAB(14); "X1"; TAB(22); "X2";
    TAB(29); "备注"; TAB(36); "|"
25  PRINT "————————————————————————"
30  FOR I= TO 5
```

```
40   READ A,B,C
50   GOSUB 1000
60   GOSUB 5000
70   NEXT I
80   PRINT "程序结束"
90   DATA 3,4,5,8,20,10,17,19,6,24,56, 43, 13,
          21,8
100    END
```

子程序—1：

```
1000 D=B*B-4*A*C
1100 IF D>0 THEN 1700
1200 IF D=0  THEN 2100
1300 X1=-B/(2*A)
1305 X1=INT(X1*10000)/10000
1400 X2=SQR(-D)/(2*A)
1405 X2=INT(X2*10000)/10000
1500 G$= "共轭虚根"
1600 RETURN
1700 X1=(-B+SQR(D))/(2*A)
1705 X1=INT(X1*10000)/10000
1800 X2=(-B-SQR(D))/(2*A)
1805 X2=INT(X2*10000)/10000
1900 G$= "不等实根"
2000 RETURN
2100 X1=-D/(2*A)
2105 X1=INT(X1*10000)/10000
2200 X2=X1
```

```
2300 G$=“相同实根”
2400 RETURN
子程序一2:
5000 PRINT“|”;TAB(3);A;TAB(6);B;TAB(9);C;
5100 PRINT TAB(12);X1;TAB(20);X2;TAB(28);
     G$; TAB(36); “|”
5200 PRINT “————————————————————————”
5300 RETURN
RUN↲           (以下是运行结果)
```

A	B	C	X_1	X_2	备 注
3	4	5	—.6667	1.1055	共轭虚根
8	20	10	—.691	—1.8091	不等实根
17	19	6	—.5589	.2016	共轭虚根
24	56	43	—1.667	.6561	共轭虚根
13	21	8	—.6154	—1	不等实根

程序结束

说明:

10～100句是主程序，主要用于提供a,b,c的值。10～25句用于打印表格的栏目。

1000～2400句是子程序一1，主要用于求解方程的根。由于我们需要把方程的系数和根用表格的形式列出，因而对X_1、X_2的值我们仅保留4位小数。采取的办法就是将X_1、X_2扩大10000倍取整，然后再缩小10000。这是由子程序一1中的1305，1405，1705，1805和2105等语句来实现的。当

然，我们也可以把对X_1、X_2保留四位小数的操作放到子程序－2中去统一处理，这样子程序－1可以更加精炼（请读者自己改进）。子程序－1中用了三个RETURN语句，这是允许的。

5000～5300句是子程序－2，主要用于打印数据和表格的横线。为了简单，横线可以用虚线。

8.1.2　编写和调用子程序时注意

1．主程序和子程序的语句行号不能交叉。主程序必须用END作为结束，子程序必须有RETURN返回主程序。参见图8.2和图8.3。

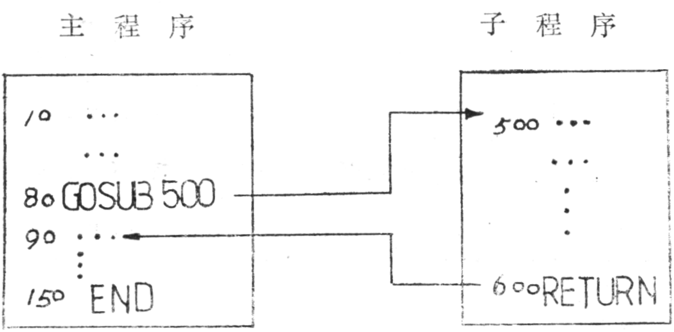


图8.2　正确的语句行号

在图8.3所示的主程序和子程序中，主程序的语句编号（200）和子程序的语句行号（200）相交了，这是不行的。

2．主程序要为子程序提供适当的数据。子程序运算的结果可以返回给主程序。

3．子程序中至少应有一个返回语句RETURN。执行

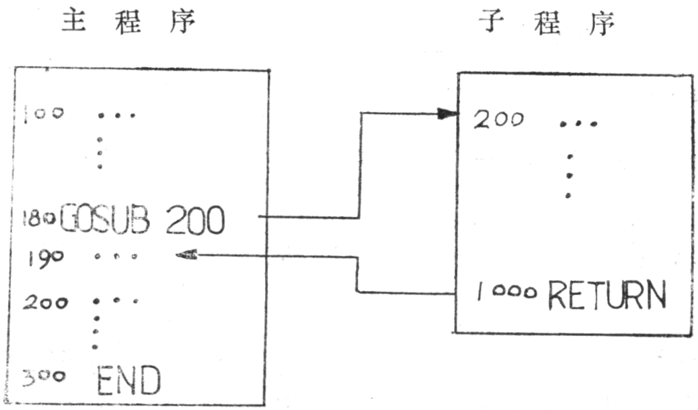


图8.3　不正确的语句行号

子程序遇到RETURN时就返回到调用语句的后继语句去执行。如果子程序中没有返回语句，子程序执行结束后就无法回到主程序相应的后继语句。

4. 调用子程序只能用转子语句GOSUB，而不能用GOTO或IF-THEN语句。因为一般调用子程序的目的是希望执行完子程序以后再返回到主程序，只有GOSUB配合RETURN才能达到这个目的。

5. 子程序可以嵌套调用。即一个子程序中又可以调用另外的子程序。这样一层一层嵌套调用下去。中华学习机CEC-I最多允许嵌套25层。

6. 主程序和子程序除了最外层的程序是地地道道的主程序以外，其余嵌套层的主程序和子程序均是相对的。即是说，某一个主程序可以是另一个程序的子程序。多重子程序的嵌套和执行情况如图8.4所示。

当然，子程—1除了调用子程序—2以外，还可以调用

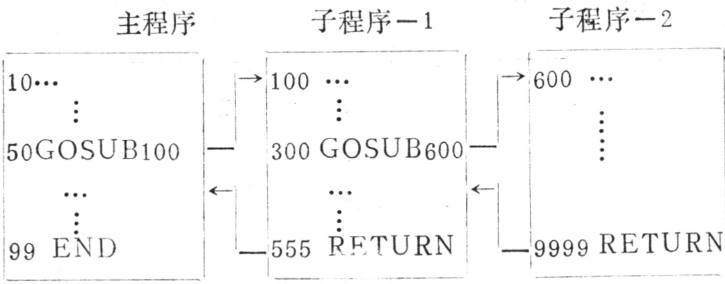


图8.4 多重子程序调用示意图

其它子程序。但一般不得递归调用。所谓递归调用是指这种情况：主程序调用子程序－1，子程序－1调用子程序－2，子程序－2又调用子程序－3，而子程序－3又反过来调用子程序－1或主程序，或者是调用自身。如图8.5所示。

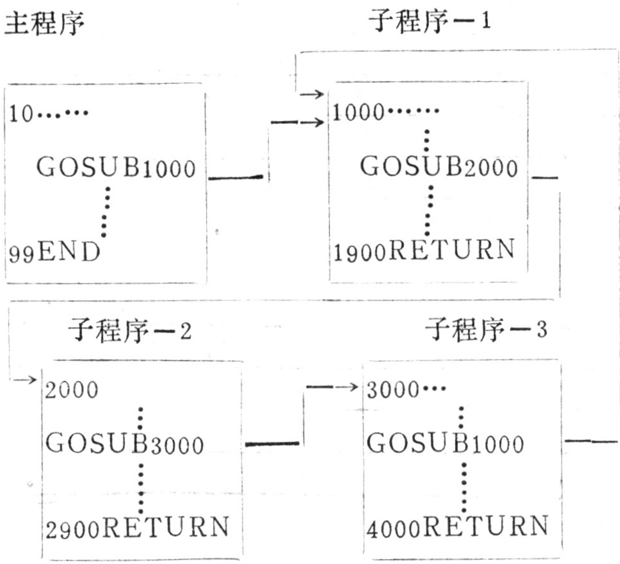


图8.4 递归调用示意图

只要不造成递归调用，子程序的调用还是比较自由的。任何子程序都不能调用最外层的主程序。我们来看一个多重调用的例子。

例8.3　子程序的多重调用。仍然用求一元二次方程根为例（若为虚根，那么X_1代表实部，X_2代表虚部）。

我们把打印表头和栏目作为子程序－1（入口是1000），求解方程的根作为子程序－2（入口是2000），打印表格作为子程序－3（入口是3000）。提供数据的部份作为主程序。

主程序：

```
10  GOSUB 1000
20  FOR I=1 TO 5
30  READ A,B,C
40  GOSUB 2000
50  NEXT I
60  PRINT "程序结束"
70  DATA 3,4,5,8,20,10,17,19,6,24,56,43,13,21,
    8
80  END
```

子程序－1：

```
1000  PRINT TAB(10); "求解方程一览表"
1002  PRINT
1005  PRINT "——————————————————————"
1010  PRINT"|";TAB(3);"A";TAB(6);"B";TAB
      (9);"C";
1015  PRINT TAB(14);"X1"; TAB(22);"X2";
      TAB(29);"备注";TAB(36);"|"
1020  PRINT "——————————————————————"
```

```
1025  RETURN
```

子程序一2：

```
2000  D=B*B-4*A*C
2005  IF  D>0 THEN 2035
2010  IF  D=0 THEN 2055
2015  X1=-B/(2*A)
2020  X2=SQR(-D)/(2*A)
2025  G$="共轭虚根"
2028  GOSUB 3000
2030  RETURN
2035  X1=(-B+SQR(D))/(2*A)
2040  X2=(-B-SQR(D))/(2*A)
2045  G$="不等实根"
2048  GOSUB 3000
2050  RETURN
2055  X1=-D/(2*A)
2060  X2=X1
2065  G$="相同实根"
2068  GOSUB 3000
2070  RETURN
```

子程序一3：

```
3000  X1=INT(X1*10000)/10000
3005  X2=INT(X2*10000)/10000
3010  PRINT"|";TAB(3);A;TAB(6);B;TAB
      (9);C;
3015  PRINT TAB(12);X1;TAB(20);X2;TAB(28);
      G$;TAB(36);"|"
```

```
3020  PRINT "————————————————————————"
3025  RETURN
RUN↵
```
（以下是运行结果。表格仅是示意。）

求解方程一览表

A	B	C	X_1	X_2	备注
3	4	5	—.6667	1.1055	共轭虚根
8	20	10	—.691	—1.8091	不等实根
17	19	6	—.5589	.2016	共轭虚根
24	56	43	—1.667	.6561	共轭虚根
13	21	8	—.6154	—1	不等实根

程序结束

程序的执行情况如下图8.5所示。

在CEC-BASIC中，还提供了一种纠错处理子程序，请参见第十章有关节段。

ON n GOSUB 行号…这种形式的转子语句用法类似ON n GOTO 行号…条件转语句。关键仍然是如何构造n。在介绍ON n GOTO〈行号〉时已经有一些例子，请读者自己改用ON n GOSUB后上机一试。

8.2 子程序实例

如果我们乐意的话，任何问题都可以用主—子程序的形式来编写。

例8.4　求$W=\frac{N!}{(A-B)!A!}$的值，其中A,B,N为已知

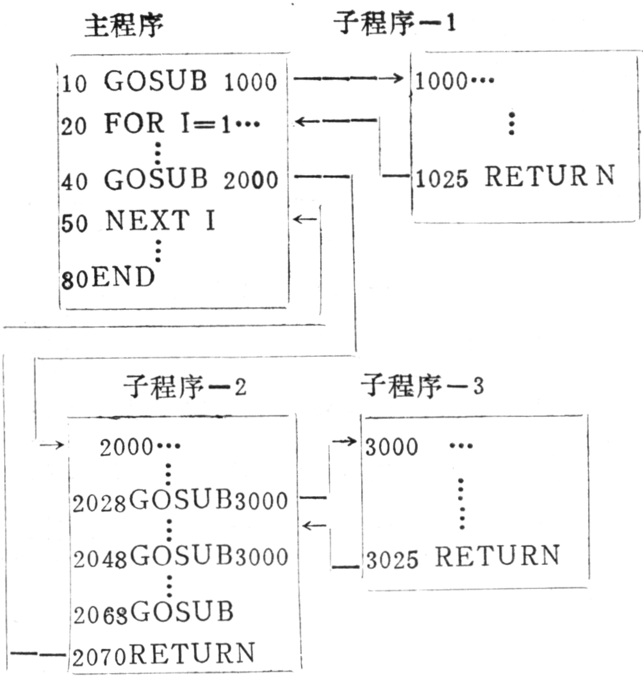
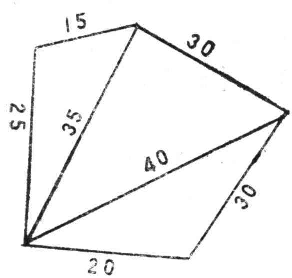


图8.5 主子程序调用示意图

数。

```
10 INPUT A,B,N
20 K=N
30 GOSUB 1000
40 W1=M
50 K=A
60 GOSUB 1000
70 W2=M
80 K=A-B
90 GOSUB 1000
```

```
100 W3=M
110 W=W1/(W2*W3)
120 PRINT "W=";W
130 END
1000 M=1
1005 FOR I=K TO 1 STEP-1
1010 M=M*I
1015 NEXT I
1020 RETURN
RUN↵
? 10,5,15
W=3003
```



在上机试用该程序时，数据不能太大，否则会发生溢出。

例8.5　求多边形的面积。

计算多边形的面积问题可以转换成求各个三角形的面积之和的问题。

求三角形面积的公式为：

$$S=\sqrt{P(P-a)(P-b)(P-c)},\quad P=\frac{a+b+c}{2},$$

其中a、b、c为三角形的三边长度。

程序如下：

```
10 W=0
20 FOR I=1 TO 3
30 READ A,B,C
40 GOSUB 1000
```

```
50 W=W+S
60 NEXT I
70 PRINT "多边形的面积=";W
80 DATA 20,30,40,40,30,35,35,25,15
90 END
1000 P=(A+B+C)/2
1010 S=SQR(P*(P-A)*(P-B)*(P-C))
1020 RETURN
RUN↵
多边形的面积=961.182579
```

例8.6 编写求两个复数进行相加和相乘的子程序，并求$W=Z_1*Z_2+Z_3*Z_4$的值。其中：

$Z_1=5+\sqrt{5}\,i$。$Z_2=\sqrt{6}-5i$，　$Z_3=3\sqrt{3}$，$Z_4=\sqrt{5}\,i$。已知两个复数相乘的公式为：

$$(a+bi)(c+di)=(ac-bd)+(bc+ad)i$$

两个复数相加的公式为：

$$(a+bi)+(c+di)=(a+c)+(b+d)i$$

两个复数相乘的子程序：

```
1000 X1=A*C-B*D
1010 X2=B*C+A*D
1020 RETURN
```

两个复数相加的子程序：

```
2000 X1=A+C
2010 X2=B+D
2020 RETURN
```

主程序：

```
10 A=5:B=SQR(5):C=SQR(6):D=-5
```

```
20 GOSUB 1000
30 W1=X1:W2=X2
40 A=3*SQR(3):B=0:C=0:D=SQR(5)
50 GOSUB 1000
60 W3=X1 : W4=X2
70 A=W1 : B=W2 : C=W3 : D=W4
80 GOSUB 2000
90 W1=X1 : W2=X2
95 IF W2<0 THEN 108
100 PRINT "W=" ;W1; "+" ;W2; "i"
102 GOTO 110
108 PRINT "W=";W1; W2; "i"
110 END
RUN↲
W=23.4277886-7.90382438 i
```

问题与思考

1. 为什么不能用GOTO转向子程序？如果一定要用GOTO转向子程序，那么可能会出现什么情况？

2.为什么子程序中必须有至少一个RETURN语句？在子程序中出现RETURN的地方用GOTO转回主程序可以吗？哪些情况下可以？哪种情况不可以？

习　　题

1. 指出下列程序运行的结果。

①
```
10 FOR I=1 TO 3
20 W= 0
```

```
  30 FOR J=1 TO 3
  40 READ A
  50 GOSUB 1000
  60 NEXT J
  70 PRINT "结果是" ; W
  80 NEXT I
  90 DATA 1,3,5,7,9,2,4,6,8
 100 END
1000 W=W+A
1100 PRINT "W=" ;W
1200 RETURN
```

②

```
  10 FOR I=1 TO 3
  15 A=I
  20 W=0
  30 GOSUB 1000
  40 PRINT "W=" ; W
  50 NEXT I
  60 END
1000 FOR J=1 TO 3
1100 GOSUB 2000
1200 PRINT "W=" ; W
1300 NEXT J
1400 PRINT "W=" ; W
1500 RETURN
2000 W=W+A
2100 RETURN
RUN↵
```

③
```
10 PRINT "主程序"
20 PRINT "AAA1"
30 GOSUB 1000
40 PRINT "AAA2"
50 GOSUB 2000
60 GOSUB 3000
70 PRINT "BBB1"
80 END
1000 PRINT "子程序－1"
1100 PRINT "CCC1"
1200 GOSUB 2000
1300 RETURN
2000 PRINT "子程序－2"
2100 PRINT "CCC2"
2200 RETURN
3000 PRINT "子程序－3"
3100 PRINT "DDD1"
3200 GOTO 1000
3300 RETURN
RUN↵
```

2. 指出程序的书写错误。

①
```
10 GOSUB 1000
20 A=5
30 GOSUB 1000
40 B=A
50 GOSUB 1000
60 C=P
```

```
    70 T=A+B+C
    80 PRINT "T=" ; T
    90 END
  1000 P=1
  1100 READ W
  1200 FOR I=1 TO W
  1300 P=(P+I)∧2
  1400 PRINT "P=" ; P
  1500 NEXT I
  1600 DATA 3, 5, 2
  1700 END
② 10 READ A,B,C
    20 FOR I=1 TO 3
    30 GOTO 1000
    40 PRINT W
    50 NEXT I
    60 PRINT "W=" ; W
    70 DATA 3,4,5
    80 END
  1000 W=A+B+C
  1100 W=W/(A+B)
  1200 PRINT A,B,C,W
  1300 RETURN
③ 10 PRINT "主程序"
    20 GOSUB 1000
    30 PRINT "AAA"
    40 END
```

```
1000 PRINT "BBB"
1100 GOSUB 2000
1200 PRINT "BBB"
1300 RETURN
2000 PRINT "CCC"
2100 GOSUB 1000
2200 PRINT "DDD"
2300 RETURN
```

④

```
  10 PRINT "主程序"
  20 GOSUB 1000
  30 PRINT "AAA"
  40 END
1000 PRINT "BBB"
1100 GOSUB 2000
1200 PRINT "BBB"
1300 RETURN
2000 PRINT "CCC"
2100 GOSUB 10
2200 PRINT "DDD"
2300 RETURN
```

3. 用子程序编写下式程序。

① $W=\frac{1-a}{1+a^2+a^5}\left(\frac{1-b}{1+b^2+b^5}+\frac{1-c}{1+c^2+c^5}\right)$

$$\frac{1-(x+y)}{1-(x+y)^2+(x+y)^5}$$

②设x、y为已知数值，求

$$y= \frac{f(x)+f(y^2)}{f(x+3)+y^3} \quad f(x+y),\ f(u)=u-\frac{u^3}{6}+\frac{u^5}{120}$$

③ $$W= \frac{f(u,v)}{f(u+v,\ u/v)+f(3u+2v,\ 3u-2v)}$$，其中：

$$f(x,\ y)= \frac{2\sin^2(x+y)-3\sin(x+y)\cos(x-y)}{e^x \ln|x|}$$

u,v为已知数值。

④ $$a=\frac{(6.9+x)^3}{x^2+\sqrt{1+2x+3x^2}} \quad ,\ x=9.5$$

$$b=\frac{(2.1z+z^4)^2}{z^4+\sqrt{1+4z^2+9z^4}} \quad ,\ z=8.24$$

$$C= \frac{(\sin y+\cos y)^3}{\sin^2 y+\sqrt{1+2\sin y+3\sin^2 y}} \quad ,\ z=4.85$$

4. 在发放工资时，财会人员不但要计算出本部门的所有职工的总工资，而且还必须把每个人的工资进行分解，以便知道需要多少张拾元卷、伍元卷、壹元卷、伍角票、贰角票、壹角票、伍分票、贰分票和壹分票。请编写一个程序，接收每个人的工资额后自动进行分解，并累计本部门发放工资时各种面额的钞票各自需要多少张。要求要用到子程序。

第九章 文件管理系统

在实际应用中，我们常常希望将一些数据、报表或程序保存起来，以备随时使用。要保存这些数据、报表或程序，就要用到中华学习机CEC-Ⅰ的主要外部设备——磁盘驱动器和软磁盘(当然也可以用录音机和磁带)。除了这些硬设备以外，还要用到中华学习机上的文件管理系统。软盘驱动器的用法是简单和直观的，第一章已经作了介绍。本章主要介绍如何使用文件管理系统在软盘上建立用户的程序文件和数据文件。

9.1 文件的基本概念

中华学习机CEC-Ⅰ把用来存放数据信息的文件都称为文本文件，因为文件是由文本字符构成的。这些文本字符可以是：运算的数据，计算的结果，待输出的报表以及BASIC程序。文本文件是通过在BASIC程序中调用操作系统的命令来建立和检索的。它可以由一个程序来建立，另一个程序来调用。

现在先介绍几个与文件有关的概念。

9.1.1 数据、字段、记录、文件

1. 数据

在介绍BASIC语句中已经涉及到了数据这个概念。但

那里对数据的说法是比较单纯的，其主要含义是指通常用到的数值常数或字符串常数。在文件管理中，我们把数据一词的含义拓广了，它不仅可以指数值常数、字符串变量的值，也可以指一个报表，一段程序。总之，我们要用到的任何符号均可以看成是数据。在文件管理中，数据可以用来表示我们所关心的任何信息。数据和信息是两个既有联系又有区别的概念。信息有一定的含义和意思。在计算机中，信息总是以数据的形式来出现。数据不一定有什么意思，它是载荷信息的一些符号。为了简单，有时不加区别的使用数据和信息这两个术语。总起来说，凡是计算机要处理的东西，我们都可以认为是数据。数据是一个包含非常广泛的术语。

2． 字段

我们先来看下面的表格：

学生情况统计表

姓　　名	学　号	性　别	年　龄	成　绩
张　山	631	男	15	95
李丽丽	632	男	13	88.5
王大力	634	女	14	74.7

在表格中列出了姓名、学号、性别、年龄和成绩共5个栏目。每一个栏目在文件管理中就称为一个字段(Field)。每一个字段下面都对应了一些数据。如姓名字段下面有张山、李丽丽和王大力三个数据。这三个数据中的每一个数据都叫姓名字段的值（实际上就是表格的每一列）。一个表格有多少个栏目（列）就有多少个不同的字段。每一个字段可

以取若干个值。总之，我们可以把表格中的栏目同字段等同起来，或者干脆就称为栏目。

3． 记录

上面的表格共有5个字段，这5个字段合起来就称为一个记录。若干字段组成一个记录。张山 631 男 15 95这是一个记录，而李丽丽 632 女 13 88.5则是另一个记录。实际上，表格中的一个横行就表示一个记录。表格中有几个横行就表示有几个记录。一个记录应当包含几个字段，这是由实际应用来确定的，当然可以很多，也可以比较少。但中华学习机CEC-Ⅰ要求一个横行（一个记录）的所有字符加起来不能超过256个。如果一个实际问题中一个记录可能超过256个字符的话，就只好分成多个表格来处理了。

从上面给出的表格我们可以看到：字段是按列来划分的，而记录则是按行来划分的。所以，以后我们又可以把字段与列、记录与行等同使用。

4． 文件

若干个记录组成一个文件。前面的学生情况统计表，它有三个记录，可以组成一个学生文件。一个文件可以同整个表格联系起来。一个文件可以代表一个表格，一个表格也可以表示一个文件。表格和文件可以等同使用。在用文件管理系统来描述文件（表格）时，首先必须给文件取一个名字，称为文件名。文件名是用字母开头的字符序列（不能含有逗号，），最多可以有30个字符，也可以直接用汉字作文件名。但必须注意，一个汉字当两个英文字符长度。因而汉字文件名最多包含15个汉字。

如果一个文件中有多个记录，那么，为了使记录有一个先后次序，因而必须给每个记录编一个顺序号，称为记录

号。记录总是从0开始编号。例如学生情况表格的记录可按如下编号：

0	张　山	631	男	15	95
1	李丽丽	632	女	13	88.5
2	王大力	634	男	14	74.7
…	…				

如果还有其它记录，可以依次编下去。这种编号可以由机器自动进行（如果由机器自动进行，那么就按输入记录的先后顺序依次编号）。

顺便介绍一下记录指针的概念。所谓记录指针，就是为了能够自由的存取文件记录而设置的一个计数器，它能够指示记录在文件中的位置。我们要存取某一个记录，首先就必须使记录指针指向该记录。根据需要，可以使指针在各个记录位置上来回拨动。一个文件打开以后，指针总是指向文件的当前记录（通常是第一个记录即0号记录），随着对文件的操作，指针位置可以不断变化。

归纳起来，我们可以得出以下结论：

①一个文件由若干个记录组成，文件名是这些记录的总代表。特殊情况下，一个文件也可以只有一个记录或没有任何记录（只有一个文件名）。

②一个记录由若干个字段组成。一个记录可以包含一个或多个字段。

③一个字段由若干个字符组成。一个字段中可以包含一个或多个字符。

④同类事物的数据可以组成一个文件。例如：所有学生的成绩档案组成一个学生文件。所有职工的工资情况组成一个工资文件。所有商品组成一个商品文件。最好不要将不同类数据组织在同一个文件中。如学生的 学 习 成绩和库存物资，这是两码事，可以分别组织成两个文件。

9.1.2 文件的种类

简单地说，中华学习机上我们经常用到的文件有三种。一是专门用于保存BASIC程序的文件，称 为 程 序文件。另两种是用于保存数据信息的文件，称为数据文件，又称文本文件。

1. 程序文件

所谓程序文件就是用BASIC语句或 DOS 命令编写的程序所组成的文件。在输入、调试、运行BASIC 程 序（特别是很长的程序）时，为了防止程序遭到意外地破坏（如：突然断电、机器故障、人为操作失误等），在调试、修改和运行之前，最好先将程序保存在软盘上，当内存被破坏时，再从软盘上把程序调到内存，重新开始调试、修改和运行。这样，不至于使我们辛辛苦苦输入到机内的程序毁于一旦。为此就要用到程序文件。

2. 顺序文件

顺序文件由字段组成，每个字段值之后用回车（CR）字符作为终结标志。当然，我们也可以用记录值来组成顺序文件，每个记录值之后同样用回车字符作为结束标志（这种情况相当于把字段值加长了）。顺序文件的数据在软盘上存放时是依次连续存放的，即一个字段（记录）接一个字段，各个字段值之间没有空格（间隙）。顺序文件的这种顺序存

贮方式可以看成是把所有数据拉成一条线而依次存放。字段和文件的长度均不受限制（当然要受到软盘容量的限制）。

顺序文件的特点是：在建立时先送入的数据放在前面，后送入的数据排列在后面。检索时也是一样，需要检索后面的数据时，必须先检索前面的数据（从第0个记录起），不管前面的数据是否需要均必须如此。例如：我们要查找第100个数据，必须先找出前面的99个数据。这如同日常排队买东西一样，先来先买。这种处理方法，用计算机的术语来说叫做“先进先出”。

顺序文件又称串行文件，其优点是：建立方式比较简单，节省存贮空间，对顺序依次存取比较快速和方便。最大的缺点是检索时必须论资排辈，依次而行，从这点来说，比较浪费时间而又不太方便。

3. 随机文件

随机文件又称直接文件。它由记录组成，每个记录中又可以包含若干字段。在随机文件中，各个记录的长度是固定的，一个记录的最大长度是32767个字符。

随机文件的特点是，建立文件时记录不一定依次连续存放，检索时可以按任意顺序检索任何记录（当然，也可以顺序检索）。文件以记录为单位进行读写操作。所谓文件的读或写是什么意思呢？写操作是指把计算机内存的数据传送到软盘的文件中（内存⇒外存）保存起来，而读操作则刚好相反，它是把外存（软盘）上文件中保存的数据传送到内存（外存⇒内存）。建立文件时传送数据必须用写（Write）操作（内存⇒外存），检索文件时传送数据必须用读（Read）操作（外存⇒内存）。

9.2 程序文件的建立和载入

如何把BASIC程序用文件的形式保存在软盘上？又如何把软盘上的BASIC程序调入内存？这是本节要解决的问题。

1. 建立BASIC程序文件的命令

格式：SAVE 〈文件名〉

作用：把内存中现有的BASIC程序用给定的文件名保存在软盘上。

例 9.1 将下列程序保存在软盘上。

```
 10 A=3:B=4:C=5
 20 W=B*B-4*A*C
 30 IF W>=0 THEN 60
 40 PRINT "方程无实根"
 50 GOTO 90
 60 X1=(-B+SQR(W))/(2*A)
 70 X2=(-B-SQR(W))/(2*A)
 80 PRINT "X1="; X1; "X2="; X2
 90 PRINT "程序结束"
100 END
SAVE E9-1↵
```

这就把10～100句的程序保存到以E9－1命名的文件中去了。（可以用CATALOG列出磁盘目录，以检查E9－1文件是否保存在软盘上）。只要我们不删除E9－1文件，它就永远保存在磁盘上。

2．载入命令

所谓载入就是把磁盘上的文件调入内存。

格式：LOAD〈文件名〉

作用：将先前保存在磁盘上的程序文件调入内存。

例 9.2 将磁盘上的E9－1文件调入内存并运行之。

```
LOAD E9-1↲
LIST↲            （显示刚调入的E9－1文件）
10 A=3:B=4:C=5
20 W=B*B-4*A*C
30 IF W>=0 THEN 60
40 PRINT "方程无实根"
50 GOTT 90
60 X1=(-B+SQR(W))/(2*A)
70 X2=(-B-SQR(W))/(2*A)
80 PRINT "X1="; X1, "X2="; X2
90 PRINT "程序结束"
100 END
RUN
```

需要注意的是：SAVE和LOAD命令只能用于建立和载入BASIC程序文件。顺序文件、随机文件是不能用SAVE和LOAD命令来建立和载入的。

3．删除文件

格式：DELETE〈文件名〉

作用：删除指定的磁盘文件。

如果一个文件确实不再需用了，可以用删除命令把它删去，从而释放出磁盘空间供其它文件使用。需要特别强调的是：一旦用该命令把文件删去，该文件就从磁盘目录中消失

了，而且不能再恢复，除非另外重新建立。所以，使用DELETE命令要格外谨慎。DELETE命令可用于删除磁盘上的任何文件（包括顺序文件、随机文件以及系统文件）。

例 9.3　删除磁盘上的E9－1文件

DELETE E9－1↵　　（删去E9－1文件）

CATALOG ↵　　（列磁盘目录）

从列出的磁盘目录中我们再也找不到E9－1文件的名字，这就表示E9－1文件已经被删去了。

9.3　顺序文件的建立和检索

9.3.1　顺序文件的建立方式

1．文件的建立或打开命令

格式：OPEN〈文件名〉[，Ss] [，Dd] [，Vv]

作用：如果文件名在磁盘目录中还没有，就表示新建一个顺序文件，机器自动将文件名列入磁盘目录中。如果文件名已经存在，则表示打开指定的文件。

OPEN命令实际上有两种可能的作用：一是建立新文件；二是打开一个旧文件。关键是文件名在磁盘目录中是否存在。如果我们取的文件名凑巧和磁盘上其它文件同名，那么仍然是作为打开文件来使用。

其中：文件名是需要建立或打开的文件的名字，它由字母开头，至多30个字符，其中不能含有逗号字符。

Ss——表示软盘驱动器所在的槽口号码，s的值可以是1～7之间的整数（但不能是3，3留作汉字系统用）。中华学习机CEC－Ⅰ上的s＝6。

Dd——软盘驱动器在盘驱接口卡上的设备号。d值可以

是1或者2。中华学习机上的d＝1。

Vv——软盘片的卷号。一张软盘有唯一的卷号，这个卷号是软盘初始化时由INIT初始化命令确定的。写上卷号的目的是防止软盘拿错。v可以是1～254之间的一个整数。v的值不能乱写，一定要根据初始化时确定的卷号，如果我们可以保证软盘不会拿错的话，Vv项完全可以省去不写。

在实际使用OPEN命令时，必须嵌入到BASIC的PRINT语句中去。

例 9.4 建立一个顺序文件，文件名就是“学生”。

```
10 D$=CHR$(4)
20 PRINT D$; "OPEN  学生, S6, D1, V254"
   ...
```

这就表示创建了一个“学生”顺序文件。为了简洁，上述语句也可以写成：

```
10 D$=CHR$(4)
20 PRINT D$; "OPEN 学生"
```

或者写成：

```
20 PRINT CHR$(4); "OPEN  学生"
```

其中：CHR$(4)可以看成是嵌入DOS命令的标志，凡是涉及到文件，都必须有CHR$(4)。OPEN命令一定要用引号”括起来并放在PRINT之后。

执行OREN命令时需要完成的工作有：

①判断文件名是否存在。如果文件名已经存在则打开文件；如果文件名不存在则创建一个新文件，将文件名列入磁盘目录中。

②分配输入/出缓冲区。

执行OPEN命令后，DOS（磁盘操作系统）给文件分配595个字节的文件缓冲区，供内外存作为交换信息之用。

③确定打开文件的最大个数是否超界。

程序中一次能够打开文件的最大个数由DOS的MAXFILE命令来确定，至多不得超过16个。如果系统没有明确指定打开文件的最大个数，那么默认个数是 3 个。

注意：①在创立或检索文件之前，必须首先打开文件。

②OPEN必须嵌套在PRINT之中。

2. 写文件的命令

格式：WRITE 〈文件名〉[，Bb]

作用：将内存的内容写入指定文件中。

其中：Bb——表示从顺序文件的第b个字节开始写数据。如果省略Bb待选项，那么就从顺序文件的第 0 个字节开始写入数据。

执行WRITE命令后，所有PRINT语句中输出的信息都输出到文件中而不是显示在屏幕上。因此，要写文件的话，WRITE命令和PRINT语句一定要联用。

例 9.5 对学生文件输入数据。

```
10 D$=CHR$(4)
20 PRINT D$;"OPEN 学生"
30 PRINT D$;"WRITE，学生"
40 PRINT"张山 631 男 15 95"
     …
```

20句——创建学生文件；

30句——写记录（或字段）到学生文件中去。

40句——将“张山 631 男 15 95”送到学生文件中作为学生文件的第 0 个字段（本例将整个记录输入）。

注意：

①WRITE命令必须嵌套在PRITE之后且用引号”括

起来。

②WRITE必须同PRINT联用。也就是说，WRITE命令写入文件的数据是由PRINT传送的。没有PRINT，数据是无法送到文件中去的。

3. 关闭文件命令

格式：CLOSE [〈文件名〉]

作用：关闭指定的文件。如果没有指定文件名，就关闭目前打开的所有文件。

一个文件操作完毕后必须及时关闭。关闭命令至少有两个作用：一是关闭文件后使操作系统收回分配的缓冲区，用于其它急待打开的文件；二是文件关闭后能够防止误操作影响文件的数据。

下面我们用具体的例子来说明如何建立一个顺序文件。

例 9.6 用下列表格中的数据建立一个顺序文件。假设顺序文件的名字就是"学生"两个汉字。

姓　名	学　号	性　别	年　龄	成　绩
张　山	631	男	15	95
李丽丽	632	女	13	88.5
王大力	634	男	14	74.7
…				

程序清单如下：

```
10 D$=CHR$(4)
20 PRINT D$;" OPEN 学生"
30 PRINT D$;"WRITE 学生"
```

```
40 FOR I=1  TO  3
50 READ A$
60 PRINT A$
70 NEXT I
80 PRINT D$。"CLOSE 学生"
90 DATA"张山 631 男 15 95"
92 DATA"李丽丽 632 女 13 88.5"
94 DATA"王大力 634 男 14 74.7"
100 END
RUN↵
```

这就把表格中的数据送到"学生"文件中保存起来了。请注意观察，当运行程序时磁盘驱动器的指示灯在不断的闪亮，这就表明内存的数据在不断地送到磁盘上。如果你希望目睹程序的运行和数据的传送情况，那么在敲入RUN命令之前，请先敲入MON C，I，O命令。这就表示在执行程序的同时又在屏幕上显示出程序的执行过程，再来试一遍。

```
MON C，I，O↵
RUN↵
OPEN 学生
WRITE 学生
张  山 631 男 15 95
李丽丽 632 女 13 88.5
王大力 634 男 14 74.7
CLOSE 学生
]
```

我们最好将以上程序保存在软盘上。

SAVE E9－6↲

我们列出目录就可以看到“学生”和E9－6两个文件都存在。

CATALOG↲

DISK VOLUME 254

T 002 学生

A 002 E9－6

注意：顺序文件只是保存了学生数据，并没有把例9.6的程序保存在其中。而程序文件则保存了例9.6的程序。千万不能混为一谈。

9.3.2 顺序文件的检索

建立文件的目的主要是为了保存信息，便于检索。

1. 读入顺序文件

格式：READ〈文件名〉[，Bb]

作用：对打开的文件进行读入操作。在执行该命令时，所有INPUT或GET语句都表示从顺序文件中接收数据而不是从键盘上接收数据。

其中：Bb——表示从顺序文件的第b个字节开始读入数据，如果省略了Bb待选项，则表示从顺序文件的第0个字节开始读入数据。

注意：

①在用INPUT接收数据时，一次只能接收读入的一个字段值。如果字段值中有逗号，则依次将各数据送到INPUT后的各个变量中。

②如果INPUT中各变量的数据类型和顺序文件中接收的数据类型不相符合，那么就给出“？Reenter”（重新输

入）信息，并继续接收下一个数据。

③如果顺序文件的字段中的数据多于INPUT语句中变量的个数，则给出信息：?EXTRAIGNORED（超过部份忽略不计），去掉多余部份。

④如果顺序文件字段中数据少于INPUT中变量的个数，屏幕显示??，系统自动到下一个字段中去读取数据。这种情况可能造成数据错位。

⑤为了防止数据错位，检索数据的格式和创建文件时的格式应该完全一致。

例 9.7 从顺序文件中读入数据。

我们以先前建立的学生文件为例，程序清单如下：

```
10 D$=CHR$(4)
20 PRINT D$;"OPEN 学生"
30 PRINT D$;"READ 学生"
40 FOR I=1 TO 3
50 INPUT B$
60 PRINT B$
70 NEXT I
80 PRINT D$;"CLOSE 学生"
90 END
RUN↲          （从学生文件中检索的数据）
张  山 631 男 15 95
李丽丽 632 女 13 88.5
王大力 634 男 14 74.7
```

2. 添加数据

格式:APPEND〈文件名〉[,Ss] [,Dd] [,Vv]

作用：打开顺序文件并允许在文件末尾添加新的数据。

其中：Ss，Dd，Vv和前面含义相同。

注意：在APPEND之后必须用WRITE命令。

例 9.8　在学生文件末尾添加两个新记录数据。

```
10 D$=CHR$ ( 4 )
20 PRINT D$; "APPEND 学生"
30 PRINT D$; "WRITE 学生"
40 PRINT "李  花 635 女 14 85.5"
50 PRINT "王  武 636 男 13 90.1"
60 PRINT D$; "CLOSE 学生"
70 END
  RUN↵
```

通过运行以上程序，就把：

李　花 635 女 14 88.5

王　武 636 男 13 90.1

这两个新记录添加到学生文件的末尾了。学生文件原来只有三个记录，现在就有五个记录。

例 9.9　取出学生文件的所有记录，用以检查新添加的数据是否添加上了。

```
10 D$=CHR$ ( 4 )
20 PRINT D$; "OPEN 学生"
30 PRINT D$; "READ 学生"
40 FOR I=1 TO 5
50  INPUT B$
60  PRINT B$
70 NEXT I
80 PRINT D$ "CLOSE 学生"
90 ENI
```

```
RUN↵
张  山  631  男  15  95
李丽丽  632  女  13  88.5
王大力  634  男  14  74.7
李  花  635  女  14  85.5
王  武  636  男  13  90.1
```

这就依次把顺序文件"学生"中的记录显示出来了。

3. 定位记录指针

格式：POSITION 〈文件名〉[，Rr]

作用：将文件的记录指针定位到指定的字段位置。

其中：Rr—表示指针指向当前位置的第r个以后（包括r）的字段。如果省去了Br待选项，就表示不移动指针。

注意：

①该命令必须作用于一个已打开的顺序文件。

②该命令必须用在WRITE和READ命令之前。

例 9.10 在学生文件中已经保存了五个记录数据（0～4号）。现在希望从学生文件中读取第三、四、五个记录。可用下列程序实现（记住：记录号从0开始）：

```
10 D$=CHR$(4)
20 PRINT D$; "OPEN 学生"
30 PRINT D$; "POSITION 学生, R2"
40 PRINT D$; "READ 学生"
50 INPUT A$, B$, C$
60 PRINT D$; "CLOSE 学生"
70 PRINT A$: PRINT B$: PRINT C$
80 END
RUN↵
```

王大力　634　男　14　74.7

李　花　635　女　14　85.5

王　武　636　男　13　90.1

第30句将指针定位在第2号记录位置上。

第40，50两句依次将第2，第3和第4号记录送到字符串变量A$、B$和C$中。

9.3.3 顺序文件的修改

严格说来，顺序文件是不能修改的。但我们可以利用WRITE命令来重写记录。

例 9.11 用"张　玲　637　女　14　93"去替换学生文件中的第一个记录。

```
10 D$=CHR$(4)
20 PRINT D$; "OPEN 学生"
30 PRINT D$; "WRITE 学生"
40 A$="张  玲  637  女  14  93"
50 PRINT A$
60 PRITT D$; "CLOSE 学生"
70 PRINT D$; "OPEN 学生"
80 PRINT D$; "READ 学生"
90 INPUT B$
100 PRINT B$
110 PRINT D$; "CLOSE 学生"
120 END
RUN↲
张  玲  637  女  14  93
```

通过执行10～60句就用"张　玲　637　女　14　93"复

盖了原来的0号记录。70～110句就将替换后的新记录显示出来了。

例 9.12　将顺序文件的第3号记录“李　花”改成“黎花”。

这可以用定位命令配合写语句来实现。

```
 10 D$=CHR$(4)
 20 PRINT D$;"OPEN 学生"
 30 PRINT D$;"POSITION 学生,R3"
 40 PRINT D$;"WRITE 学生"
 50 A$="黎　花　635　女　14　85.5"
 60 PRINT A$
 70 PRINT D$;"CLOSE 学生"
 80 PRINT D$;"OPEN 学生"
 90 PRINT D$;"READ 学生"
100 FOR I=1 TO 5
110 INPUT B$
120 PRINT B$
130 NEXT I
140 PRINT D$;"CLOSE"
150 END
 RUN↲
张　玲　637　女　14　93
李丽丽　632　女　13　88.5
王大力　634　男　14　74.7
黎　花　635　女　14　85.5
王　武　636　男　13　90.1
```

第30句将记录指针定位在第3号记录（实际是文件中的

第四个记录）上，第40～60句表示写文件，这时就把A$中的内容通过60句写入到顺序文件中（复盖了原来的记录）。

第80～140句将学生文件的五个记录全部显示在屏幕上，由此看出，修改是成功的。

对顺序文件，归纳起来必须注意以下几点：

①不论是建立、检索或修改文件都必须先打开(OPEN)文件（APPEND命令除外）后使用（建立）。

②建立和修改文件时，OPEN、WRITE、PRINT三者必须联用。（内存送外存）

③检索、打印文件记录时，OPEN、READ、INPUT三者必须联用。（外存送内存）

④对于定位命令，如果定位是为了检索、打印记录，那么OPEN、POSITION、READ、INPUT四者必须联用；如果定位是为了修改记录，那么OPEN、POSITION、WRITE、PRINT四者必须联用。

⑤文件使用以后必须关闭。

⑥一个新建的文件必须先关闭再重新打开后才能马上进行检索。不能够一边建立一边检索。

⑦顺序文件只能按顺序建立和存取。

9.4 随机文件的建立和检索

9.4.1 随机文件的建立方式

1. 随机文件的建立命令

格式：OPEN 〈文件名〉Lj [,Ss] [,Dd] [,Vv]

作用：建立或打开一个随机文件。如果文件名不存在，就新建立一个文件并把文件名列入磁盘目录中；如果文件名

已经存在，就打开该文件。

其中：

Lj——表示记录的长度。j可以是1～32767 之间的一个整数。

Ss，Dd，Vv含义同前述。

在用OPEN命令来建立随机文件时，DOS 并不把记录的长度Lj存入磁盘中。如果打开文件时指定的记录长度与建立文件时的长度不一致，那么DOS始终按现在给出的记录长度来计算文件中各个记录的位置。如果存在前后记录长度不一的现象，读取记录时可能造成错位。为避免出错，创建文件时的记录格式和检索记录时的格式应当完全一样。

例 9.13 创立一个随机文件。

我们仍然以学生情况统计表格为例来说明如何建立随机文件。

```
10 D$=CHR$(4)
20 PRINT D$;"OPEN 学生-L25，L25"
```

为了使随机文件名不同于顺序文件名，这里把随机文件的名字取为“学生-L25”，表示学生文件的记录长度是25个字符。L25就表示了记录长度。

2. 写随机文件

格式：WRITE 〈文件名〉[，Rr] [，Bb]

作用：将记录写到指定的随机文件中去。

其中：

Rr——表示指定文件的第r个记录，r的值可以是0～32767之间的一个整数，这个数就表示把记录写到随机文件的第几号记录。当r＝0或省略不写时，就表示从随机文件的第0号记录开始写入记录内容。从这里可以看到，随机

文件并不象顺序文件那样必须顺序写入，而是可以按任何次序（由用户确定）来建立文件。当然，不按顺序来建立就必须指出记录号码。这就是参数Rr的作用。

Bb——表示从该记录的第b个字节开始执行写操作（即写入的内容从第b个字节开始存放）。如果b值太大以致超过了记录划定的长度，那么可能影响到其它记录的存放位置，从而导致数据和记录位置错位。

在使用WRITE来写随机文件时，需要注意：

①使用WRITE之后，在没有关闭文件之前，不能用DOS的其它命令或INPUT、GET语句，否则，将会停止WRITE命令的执行。

②尽管随机文件记录的写入可以不按顺序，但我们最好还是依次顺序写入，以免弄错或弄掉一些记录。

例 9.14　随机文件的写入命令用法示例。

```
10 D$=CHR$(4)
20 PRINT D$; "OPEN 学生_L25, L25"
30 PRINT D$; "WRITE 学生_L25"
```

第20句创立一个随机文件"学生_L25"。

第30句表示从随机文件的第0个记录位置开始写入记录。

在建立随机文件时，除了在PRINT中嵌套OPEN和WRITE命令以外，还需要有PRINT语句的配合，才能完成文件的建立工作。下面我们仍然使用学生情况统计表中的数据来建立一个完整的随机文件。

例 9.15　建立随机文件的完整示例。

```
10 D$=CHR$(4)
20 PRINT D$; "OPEN 学生_L25, L25"
```

```
30 FOR I=1 TO 3
40 PRINT D$; "WRITE 学生_L25, R"; I
50 READ A$, B, C$, D, E
60 PRINT A$, B, C$, D, E
70 NEXT I
80 PRINT D$; "CLOSE 学生_L25"
90 DATA 张  山, 631, 男, 15, 95
92 DATA 李丽丽, 632, 女, 13, 88.5
94 DATA 王大力, 634, 男, 14, 74.4
100 END
MON C, I, O
RUN
OPEN 学生_L25, L25
WRITE 学生_L25, R1
张  山          631          男
                15           95
WRITE 学生_L25, R2
李丽丽          632          女
                13           88.5
WRITE 学生_L25, R3
王大力          634          男
                14           74.7
CLOSE 学生_L25
]
```

从以上显示的运行情况看出，我们需要的数据确实写入了随机文件"学生_L25"中。需要说明的是：

第20句表示打开随机文件，文件名字是"学生_L25"，

L25表示随机文件的记录长度。打开文件必须按20句给出的形式书写。

第40句表示把数据写入文件。其中引号中的“R”表示记录的意思，分号；后的“I”用以标明是哪个记录（即记录号）。写文件的形式也必须如此。

第50句是将数据区的数据传送给变量A$、B、C$、D、E。第60句才真正将数据传送到软盘上的文件中去。

第90～94句用于提供数据。从例中看到，字符型数据也可以不带引号”。

要完成对随机文件的写入操作，也必须WRITE配合以PRITE语句。关闭文件的命令CLOSE总是需要的，格式同前述一样。

9.4.2 随机文件的检索

检索任何文件，均需要将其读入到内存，随机文件也不例外。现在看看随机文件的读入命令：

格式：READ〈文件名〉[,Rr] [,Bb]

作用：将指定文件的指定记录读入到内存。

其中：

Rr——表示对随机文件中的第r个记录执行读入操作。r的值一定是随机文件的记录号。

Bb——表示从记录的第b个字节开始读入。

我们先来看下面的例子：

例 9.16 把学生_L25随机文件的第3个记录读入到内存，并在屏幕上显示出来。

```
5 INPUT “I=”；I
10 D$=CHR$（4）
```

```
20 PRINT D$; "OPEN 学生_L25, L25"
30 PRINT D$; "READ 学生_L25, R"; I
40 INPUT A$, B, C$, D, E
50 PRINT D$; "CLOSE 学生_L25"
60 PRINT A$; ""; B; ""; C$; ""; D; ""; E
70 END
  RUN↵
I= 3 ↵
王大力  634  男  14  74.7
```

如果我们希望看到程序的执行过程，运行之前先敲入监控命令MON，如下所示：

```
MON  C, I, O↵
RUN↵
I= 3 ↵
OPEN 学生_L25, L25      （表示正在打开学生_L25
                          文件）
READ 学生_L25, R3       （正在读第3个记录，即
                          R3的意思）
? 王大力
? ? 634
? ? 男
? ? 14
? ? 74.7
CLOSE 学生_L25
王大力  634  男  14  74.7
]
```

上面程序中的第40和60两句可以换成只需一个变量的情

况：

```
40 INPUT A$
60 PRINT A$
```

运行结果如下：

```
MON C, I, O↲
RUN↲
I=1↲
OPEN 学生_L25, L25
READ 学生_L25, R1
? 张  山 631 男 15 95
CLOSE 学生_L25
张  山631男1595
```

这种情况输出的结果彼此间没有空格，不大容易阅读。

9.4.3 随机文件的添加和修改

随机文件的添加和修改都比较方便，我们要添加和修改哪个记录，只要给出记录号重新写入就行了。我们用例子来说明。

例 9.17 在学生文件中添加两个新记录。

4号 李 花 635 女 14 85.5

5号 王 武 636 男 13 90.1

程序如下：

```
10 INPUT "I=" ; I
20 INPUT A$, B, C$, D, E
30 D$=CHR$(4)
40 PRINT D$; "OPEN 学生_L25, L25"
50 PRINT D$; "WRITE 学生_L25, R" ;
```

```
60 PRINT A$, B, C$, D, E
70 PRINT D$; "CLOSE 学生_L25"
80 END
RUN↲
I=4↲
? 李  花, 635, 女, 14, 85.5↲
OPEN 学生_L25, L25
WRITE 学生_L25, R4
李  花          635        女
                14         85.5
CLOSE 学生_L25
RUN↲
I=5↲
? 王  武, 636, 男, 13, 90.1↲
OPEN 学生_L25, L25
WRITE 学生_L25, R5
王  武          636        男
                13         90.1
CLOSE 学生_L25
]
```

我们来看看学生_L25文件中是否添加了两个新记录（4号和5号两个记录）。

例 9.18 检查学生_L25文件中是否添加了两个记录。

```
10 D$=CHR$(4)
20 PRINT D$; "OPEN 学生_L25, L25"
30 FOR I=1 TO 5
40 PRINT D$; "READ 学生_L25, R"; I
```

```
50 INPUT A$
60 PRINT A$
70 NEXT I
80 PRINT D$; "CLOSE 学生_L25"
90 END
RUN↵
张  山 631 男 15 95
李丽丽 632 女13 88.5
王大力 634 男 14 74.7
李  花 635 女 14 85.5
王  武 636 男 13 90.1
]
```

例 9.19 将1号记录换成：张 玲 637 女 14 93

```
10 INPUT "I="; I
20 INPUT "输入修改数据：" A$, B, C$, D, E
30 D$=CHR$(4)
40 PRINT D$; "OPEN 学生_L25 L25"
50 PRINT D$; "WRITE 学生_L25, R", I
60 PRINT A$, B, C$, D, E
70 PRINT D$; "CLOSE 学生_L25"
80 END
RUN↵
I=1↵
输入修改数据：张  玲, 637, 女, 14, 93↵
OPEN 学生_L25, L25
WRITE 学生_L25, R1
张  玲          637          女
```

14　　93

我们要检查文件中的修改是否属实，可以调用例9.18的程序。运行结果如下：

RUN↵

张　玲 637 女 14 93

李丽丽 632 女 13 88.5

王大力 634 男 14 74.7

李　花 635 女 14 85.5

王　武 636 男 13 90.1

对于随机文件来说，我们要注意以下几点：

①建立文件、修改和添加记录时都是OPEN、WRITE、PRINT三者联用。

②检索文件时，OPEN、READ、INPUT 三者联用。

③无论是写（内存送外存）还是读（外存送内存）操作，均要提供记录号。机器就是根据记录号来进行直接存取的。当然，记录号并不要求一定依次连续，但不能超界。记录号的范围是 0 ～32767。在建立文件时，如果乐意的话，完全可以先写入 5 号记录，再写入 3 号记录。但这样交错进行的缺点是有时候自己都不清楚写入了哪些记录。因此，建议建立文件时依次顺序写入记录，文件建好以后随机检索和修改记录。

④随机文件中不能用POSITION和APPEND命令。

⑤建立和检索文件之前用OPEN打开文件，用后用CLOSE关闭文件。

⑥所有命令都必须嵌入PRINT之中才能生效。

问题与思考

1． 文件是若干数据的集合，数组也是若干数据的集合，两者的区别是什么？

2． 顺序文件和随机文件最根本的区别是什么？

3． 为什么随机文件不用POSITION和APPEND命令？

习 题

1． 分别用顺序文件和随机文件来建立如下的人事文件：

代号	姓名	性别	年龄	职称	基本工资
84315	李顺	男	48	工程师	129.4
…	…	…	…	…	…

并对文件进行增加和修改，写出相应的程序。

2． 建立一个工资管理文件系统，该系统能够建立工资文件，增添新职工工资信息，修改职工工资，统计所有职工的总工资。写出进行这些操作的程序。

第十章　BASIC程序的调试与设计技术

到上一章结束为止，我们已介绍了BASIC 语言各语句的使用规则、各函数的功能以及运用这些语句和函数来编写程序的一些方法。在这一章中，将讨论有关程序设计与调试的一些技术。配以实例分析，以求进一步巩固已学的内容，提高程序设计与调试的技能。

10.1　程序的调试

人们往往有这样一种想法：认为程序编写完成以后，则大功告成了，但是事实并非如此。对于稍大一点的程序，大部分工作量是花在程序编写出来之后，即调试和维护阶段，其中调试的工作量约占开发期的一半。从某种意义上来说，调试和测试一个大型程序系统的难度，可能比设计这个程序系统还要大。为此，BASIC语言提供了程序跟踪（TRACE）与取消（结束）跟踪（NOTRACE）语句，用于调试程序时侦错。

10.1.1　追踪程序的运行

格式：TRACE

作用：跟踪程序的执行，将执行过的行号及运行结果均显示或打印出来。

例10.1　TRACE语句实例。（假定我们希望追踪程序

的执行，了解程序的执行情况，可用TRACE语句。

```
5  TRACE
10 FOR B=5  TO 10
20 PRINT B, B*2
30 NEXT B
40 END
RUN↙
# 5     #10    #20    5     10
        #30    #20    6     12
        #30    #20    7     14
        #30    #20    9     18
        #30    #20    10    20
        #30    #40
```

说明：本程序中行号为5的语句为追踪（TRACE）语句，故用RUN命令运行时，不仅给出计算结果，还按运行路径给出语句行号。其中带#号者为执行的行号，不带#号者为给出的计算结果。

10.1.2 取消追踪

格式：NOTRACE

作用：结束（取消）TRACE语句的跟踪功能，即停止跟踪程序的执行。

注意：只有NOTRACE才能取消TRACE的作用。RUN、CLEAR、NEW、DEL与RESET都不能解除TRACE的作用。

例10.2 TRACE和NOTRACE语句实例。

```
5  TRACE
```

```
10 FOR B=5  TO 10
20 PRINT B, B*2
30 NEXT B
40 NOTRACE
50 FOR F=5 TO 10
60 PRINT F, F*2
70 NEXT F
80 END
RUN↵
# 5       #10      #20      5       10
          #30      #20      6       12
          #30      #20      7       14
          #30      #20      8       16
          #30      #20      9       18
          #30      #20      10      20
          #30      #40      5       10
                            6       12
                            7       14
                            8       16
                            9       18
                            10      20
```

说明：程序中行号为40的语句为取消追踪语句，当运行到该行以后的语句时不再给出行号。

10.2 出错处理语句

格式：ONERR GOTO〈行号〉

作用：程序发生错误后转到指定行号的子程序去进行处理。

格式：RESUME

作用：从纠错处理子程序返回到主程序。

说明：在程序执行过程中，一旦发生错误，计算机就转到由语句〈行号〉指定的"纠错处理"子程序入口执行。RESUME是"纠错处理"子程序的返回语句。系统内存216单元作为出错标志单元，若该单元的B7位为1，则表示有错误发生。发生错误后，出错类型码存放在222单元。下表为中华学习机BASIC语言出错类型编码表：

类型码	出错情况
0	NEXT与FOR不匹配
16	句法错
22	RETURN与GOSUB不匹配
42	缺少数据
53	非法量
69	溢出
77	缺少内容
92	未定义语句
107	不适当的下标
120	重定义数组
133	除数为零
163	类型不匹配
176	串太长
191	形式太复杂
224	未定义函数
254	响应INPUT语句时转入不适当

255　　用CTRL－C中断纠错程序

下面的程序说明ONERR GOTO语句的用法及"纠错"子程序的编写方法。此程序处理了"非法量"与"除数为零"两种类型的错误。为了帮助理解，将程序的执行过程用TRACE语句记录下并显示出来。

```
  5 TRACE
 10 ONERR GOTO 100
 20 T=-5
 30 A$=LEFT$("EDUCATION COMPUTE
      R", T)
 40 PRINT A$
 50 S=300
 60 ONERR GOTO 100
 70 A(5)=S/A(0)
 80 PRINT "A(5)="; A(5)
 90 END
100 POKE 216, 0
110 PRINT "PEEK(222)="; PEEK(222)
120 IF PEEK(222)=53 THEN T=9
130 IF PEEK(222)=133 THEN A(0)=5
140 RESUME
]   RUM↵
 #10   #20   #30   #100  #110  PEEK(222)=53
 #120  #130  #140  #30   #40   EDUCATION
 #50   #60   #70   #100  #110  PEEK(222)=133
 #120  #130  #140  #70   #80   A(5)=60
 #90
```

程序中第10行令其出错后转100行语句处理错误，当从第5行执行到第30行出现错误，则按10行的指令转到100行执行，将出错标志单元216置“0”，取消出错标志，然后顺序执行110句，显示错误类型码值53，为非法变量错。执行120句条件为真，将T赋值9，执行130句条件不成立，顺序执行140句因为140句是100～140句这个纠错处理子程序的返回语句。它的作用是返回到出错的语句去执行，故执行30句，将字符串中左起1～9个字符送入A$，执行40句打印出EDUCATION。执行50句将S赋值300，执行第60句又记下出错转100语句，执行70句A(0)之值为零，又转到100句执行，将出错标志单元置“0”，执行110句，打印存放出错类型码单元222之值133，执行120句条件为假，执行130句条件为真A(0)赋值5，执行140句返回到70句，将S除以A(0)之值(300/5=60)送入A(5)，执行80句打印A(5)之值60，最后执行90句结束。

注意：ONERR GOTO的作用是诊断出错，并转到相应行号的子程序去处理。而RESUME只能是纠错处理子程序的出口语句，其作用是返回到出错的语句行号并重新执行之。

10.3 节省存贮空间

10.3.1 节省存贮空间的一些措施

为了使一个程序适合于在配置较小的内存机器系统上运行，下面的提示是会有用的。然而，其前面两条节省内存空间的方法仅当面对严重的内存紧张的情况下采用。较好的方法是保留两种程序版本，一种是带有大量文字说明（REM

注释句），另一种是压缩（即不含注释语句）的版本。前者利于阅读修改，后者节省内存。

1. 每一行上使用多个语句，使得只有少量的空间（5个字节）与程序中的每一行相关连。这5个字节中的2个字节以2进制形式存放了该行的行号。这意味着，程序中的行号不论有多少数字（0≤行号≤65529），均占用2个字节。即是说减少一个行号就可以少占两个字节的内存。一个独立行最多可以有239个字符。

2. 删除所有的REM语句，每个REM语句使用的字节数至少为一个字节加上通常的正文的字节数。（REM是注释语句，没有多大用处，故BASIC语言中略去了）。例如：语句170REM THIS IS A COMMENT占用了24个字节的内存。

3. 尽可能地用整型数代替实型数。

4. 用变量代替常量,假定一个程序中使用常量3.14159十次，如你的程序中插入一条语句10 P=3.14159，并在每次需要3.14159的地方用P代替，如此可节省40个字节，并可提高程序运行效率。

5. 一个程序不一定要用END来终止，因此，程序末尾的END语句可以删除。

6. 重复使用同一个变量。如某程序中在前部分用变量或数组T保留临时结果，则在这个程序的后面部分需要临时变量时仍用变量或数组T。

7. 用自定义函数及子程序减少完成相同作用的程序段。

8. 使用数组的零元素。例如：x（0），y（0，2）。

9 当A$=“CAT”又被赋值为A$=“DOG”时，

原来的串“CAT”不从存储器中抹去。在程序中定期使用形如X＝FRE（0）的语句，将使CEC－BASIC从存储的顶部开始“内部清洗”老的字串。

10.3.2 BASIC的存储分配情况

简单的（非数组）实型、整型或变量，象V，V%或V$使用7个字节。实型变量用2个字节作为变量名，用5个字节作为其值（1个指数，4个尾数）；整数变量用2个字节作为变量名，2个字节作为其值，其余3个字节为0；串变量用2个字节作为变量名，1个字节为串的长度，2个字节为指向内存中串的位置的指针，其余2个字节为0。

实型数组变量最少使用12个字节：两个字节为变量名，两个字节为数组的大小，一个为维数，每一维的大小使用两个字节，每个数组元素使用5个字节。整型数组变量的每个整数元素使用两个字节。串数组变量的每个元素使用三个字节：一个字节作为长度，二个字节作为指针。

串变量，无论简单的还是数组，对串中的每个字符使用内存的一个字节。串本身是按照在程序中的先后次序来定义的，从HIMEM：开始存放。

当使用一个DEF语句定义一个新的函数时，用6个字节来存储指向定义的指针。

保留字如FOR，GOTO及内部函数名，如COS，INT及STR$仅占程序存储的一个字节，程序中其它字符，每个字符占一个字节。

当一个程序开始执行时，在栈中空间是按下列规则动态分配的：

1．每个有效的FOR…NEXT循环用16个字节。

2．每个有效的GOSUB（还没有执行到RETURN）用6个字节。

3．对一个表达式中的每对圆括号用4个字节，以及表达式中的每个计算出的结果用12个字节。

10.4 提高程序的执行速度

本节的提示可以提高中华学习机BASIC程序的执行速度。

1．使用变量代替常量。把一个常量转换成浮点（实型数）表示要比读取一个简单变量或数组变量花费的时间更多。尤其是在FOR…NEXT循环内部或其它反复被执行的编码中这是非常重要的。

2．在BASIC程序的执行期间，首先遇到的变量是从变量表的初始位置开始分配的变量。这个意思是，如一个语句。

```
10      A=0:B=A:C=A
```

BASIC在变量表中第一个位置放A，第二个位置放B，第三个位置放C（假定第10行是程序中的一个执行语句）。在本句后面的程序中，当BASIC碰到一个引用变量A时，只搜索变量表的第一项就找到了A，搜索第二项可找到B，搜索第三项可找到C等等。

3．采用不带循环变量的NEXT语句，NEXT比NEXTI稍快一点，因此不要检查在NEXT中指出的变量是否与最近的仍然有效的FOR语句中的循环变量为同一变量。

4．程序执行期间，当BASIC遇到引用一个新行时。例如：1000　　GOTO　　10000

它将从起始行开始扫描整个用户程序，直到找出所要引用的行号为止（在这个例子中是10000），因此，经常引用的行在程序中尽可能地放前一些。

5. 在程序运算中若以加法代替乘法，以乘法代替乘方，可以加快运算速度。

6. 在循环结构中减少多重嵌套的层数或减少循环的次数对于减少运算时间都是有利的；对于与循环体内循环变量无关的表达式一律提到循环之外，可避免重复计算同一个式子。

10.5 程序的链接运行

链接操作的目的是装入运行第二个BASIC程序时，第一个程序运行的变量结果仍保留在内存中，可以为第二个程序所使用。这相当于把两个小程序链接成了一个大程序。如何实现两个程序链接运行呢？在DOS3.3系统主盘上提供了CHAIN程序，利用该程序，可以完成BASIC程序之间的链接。使用时首先将CHAIN程序复制到我们的工作盘上，然后在第一个程序的末尾加上以下两条语句即可。

```
PRINT CHR$(4);"BLOAD CHAIN,
   A520"
CALL 520"第二个程序名"
```

设我们有如下两个程序：

程序AA：

```
5 DIM A(10)
10 FOR I=1 TO 5:A(I)=I*2:PRINT A(I),
   :NEXT I:PRINT
```

```
20 PRINT "CHAIN AB"
30 END
] RUN↵
…
```

程序AB：

```
10 FOR J=1 TO 5:A(J)=A(J)+J:
    PRINT A(J), :NEXT J:PRINT
20 END
] RUN↵
```

我们将这两个程序按新要求改正如下，并保存在工作盘上。（然后运行第一个程序，即实现AA和BB两个程序的链接运行）。

程序AA：

```
5  DIM A(10)
10 FOR I=1 TO 5:A(I)=I*2:PRINT A(I),
   :NEXT I:PRINT
20 PRINT CHR$(4); "BLOAD CHAIN,
       A520"
30 CALL 520 "AB"
40 END
```

程序AB：

```
10 FOR J=1 TO 5:A(J)=A(J)+J:PRINT
       A(J), :NEXT J:PRINT
20 END
```

打入RUN AA命令后，屏幕将显示：

```
] RUN↵
2          4          6
```

8	10	
3	6	9
12	15	

10.6 BASIC应用程序设计实例

计算人体（生物）周期

根据生物周期学理论，一个人的三个生物周期是：

体力周期——23天

情绪周期——28天

智力周期——33天

对任给一个日期的三个周期振幅值计算公式如下：

振幅值＝SIN（360×自出生到所给日期总天数／周期天数）

要求编写一个程序完成对任给日期之三个振幅值及其平均值的计算。

（一）设计思路

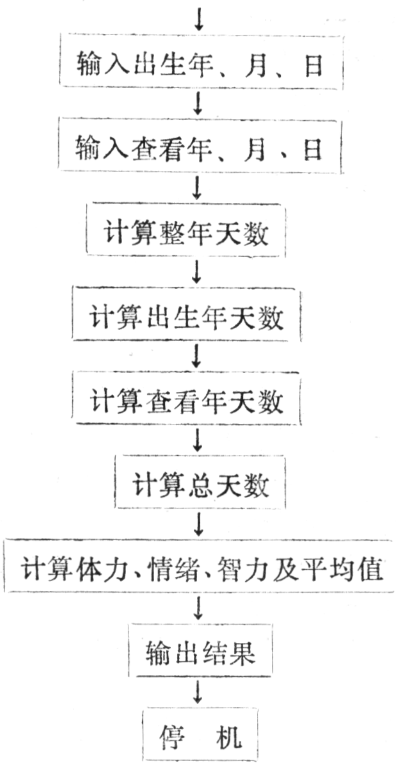
本题需把某人的生日（包括出生年月）和所查日期（包括年月）作为已知数据输入，并求出从生日到查看日期的总天数，再将其与三个周期（天数）值分别代入振幅值计算公式，完成振幅值计算。

求出从生日到查看日期的总天数，可以分为三步：第一步求全部整年的天数；第二步求出生年份的天数；第三步求查看年份的天数。

其中整年天数为各个整年天数之和。由天文历法知在1900～2000年之间能被4整除的是润年，2月为29天，否则2月为28天，故润年为366天，平年为365天；出生年份的天

数可用从出生月到年底之和，再减去出生月生日之前的天数的方法计算；查看年份的天数可用占有整月的天数与查看月天数相加求出。以上三个部分和相加所得结果为自出生至查看之日的总天数。最后将总天数及周期天数代入公式计算并输出。

（二）框　图



（三）源程序及运行结果

10 REM这是一个计算人体（生物）周期的程序

20 INPUT“请输入出生年号：××××”；N_1

```
30 INPUT "请输入出生月号：××"；Y1
40 INPUT "请输入出生日期：××"；R1
50 INPUT "请输入查看年号：××××"；N2
60 INPUT "请输入查看月号：××"；Y2
70 INPUT "请输入查看日期：××"；R2
80 ZNTS=0
90 N1N2=N2-N1-1
95 IF N1N2<1 THEN 160
100 FOR I=1 TO N1N2
110 N=N1+1
120 RN=N/4-INT(N/4)
130 IF RN=0 THEN ZNTS=ZNTS+366
140 IF RN<>0 THEN ZNTS=ZNTS+365
150 NEXT I
160 SNTS=0
165 RN=N1/4-INT(N1/4)
170 FOR I=Y1 TO 12
180 IF I=1 OR I=3 OR I=5 OR I=7 OR I=8
    OR I=10 OR I=12 THEN SNTS=SNTS+31
190 IF I=4 OR I=6 OR I=9 OR I=11 THEN
    SNTS=SNTS+30
200 IF I=2 AND RN=0 THEN SNTS=SNTS
    +29
210 IF I=2 AND RN<>0 THEN SNTS=SN
    TS+28
220 NEXT I
230 SNTS=SNTS-R1+1
```

```
240 CNTS=0
245 RN=N2/4-INT(N2/4)
250 FOR I=1 TO Y2-1
260 IF I=1 OR I=3 OR I=5 OR I=7 OR I=8
    OR I=10 OR I=12 THEN CNTS=CNTS+
    31
270 IF I=4 OR I=6 OR I=9 OR I=11 THEN
    CNTS=CNTS+30
280 IF I=2 AND RN=0 THEN CNTS=CNTS
    +29
290 IF I=2 AND RN<>0 THEN CNTS=CNTS
    +28
300 NEXT I
310 CNTS=CNTS+R2
320 ZTS=ZNTS+SNTS+CNTS
330 TL=SIN(360*ZTS/23)
340 QX=SIN(360*ZTS/28)
350 ZL=SIN(360*ZTS/33)
360 PJ=(TL+QX+ZL)/3
370 PRINT "体力"; TL
380 PRINT "情绪"; QX
390 PRINT "智力"; ZL
400 PRINT "平均"; PJ
410 END
] RUN↵
请输入出生年号：××××1947↵
请输入出生月号：××4↵
```

请输入出生日期：××24↵

请输入查看年号：××××1988↵

请输入查看月号：××8↵

请输入查看日期：××5↵

体力　　　　.93

情绪　　　　.55

智力　　　　.44

平均　　　　.64

（四）附注：

①第20至70语句提示输入出生及查看年、月、日；

②第80至150语句计算占有的整年合计天数；

③第160至230语句计算出生年中的合计天数；

④第240至310语句计算查看年中的合计天数；

⑤第320语句计算自出生到查看日期的总天数；

⑥第330至360语句计算体力、情绪、智力及其平均值；

⑦第370至400语句输出计算结果。

⑧运行中的命令及输入用下画线以区别。

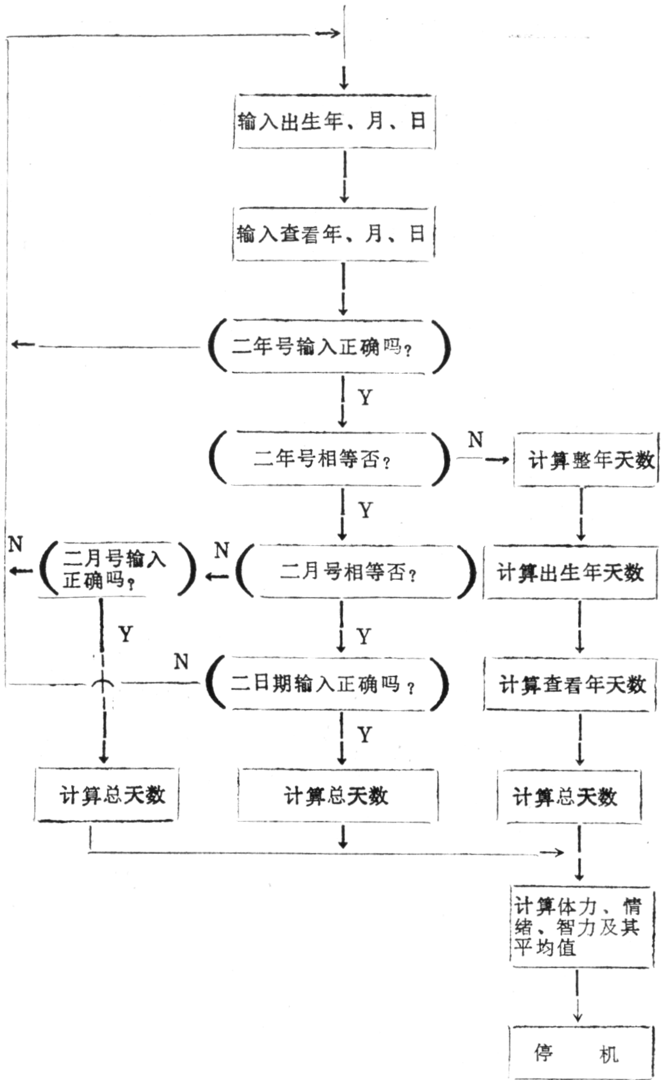
⑨本程序没有考虑输入错误及二年号相同的处理。有兴趣的同志可在本程序基础上扩充实现。于此提供其框图供参考(见下页)。

习题

1．TRACE有何作用？用什么语句能够解除TRACE状态？

2．ONERR GOTO有何作用？它必须和什么样的程序联合使用？

3．10.2节中的ONERR GOTO的例子中是否可省掉



输入出生年、月、日
输入查看年、月、日
二年号输入正确吗？
Y
二年号相等否？
N
计算整年天数
Y
N
二月号输入正确吗？
N
二月号相等否？
计算出生年天数
Y
Y
N
二日期输入正确吗？
计算查看年天数
Y
计算总天数
计算总天数
计算总天数
计算体力、情绪、智力及其平均值
停　　机

90句？

4. 试编写一个含有ONERR GOTO的程序并上机通过。

5. 试编写一个用加法代替乘法、用乘法代替乘方的程序。

6. 参考10.6中计算人体生物周期程序末尾(即263页)给出的框图，编写一个比较完善的计算人体生物周期的程序并上机运行。

第十一章 BASIC绘图和音乐

本章主要介绍中华学习机上BASIC的图形、发音功能及其程序设计方法。中华学习机BASIC具有低分辨图形及高分辨图形两种图形显示方式。这两种图形方式的主要差别，在于屏幕显示的点的大小和数量不同。“低分辨率图形”使用的点要比“高分辨率图形”的大。因而能在屏幕上显示的点数较少，在低分辨率图形的屏幕上可同时得到16种颜色，高分辨率图形下可同时得到 8 种颜色。

11.1 BASIC低分辨率绘图语句

1．GR

格式：GR

作用：将屏幕设定为低分辨率图形显示方式。

在屏幕的上部是40×40的图形区域，下部留 4 行作为文本显示区。这种显示方式称为低分辨率图象与文本的混合显示方式。

GR命令将屏幕上的图象显示区置为黑色，即具有清除图象画面的作用。

从图文混合方式转入全屏纯低分辨图象显示（不留下部 4 行为文本显示区），使用置全屏低分辨图形指令：POKE－16302，0

从全屏纯低分辨图象显示方式转到图文混合显示方式用返回图文混合型指令：POKE－16301，0

2． COLOR

格式：COLOR＝n

作用：置低分辨率图形的颜色。

其中n从0—15（共16种颜色）中取值。如果n为实数，则自动将其转换为整数。n取值大于16，小于等于255，自动以16为模处理。n值与颜色的对应关系如下：

n值	颜色
0	黑
1	红
2	深蓝
3	紫
4	深绿
5	灰
6	中蓝
7	浅蓝
8	棕
9	橙
10	灰绿
11	粉红
12	浅绿
13	黄
14	绿蓝
15	白

3． SCRN 函数

格式：SCRN（X，Y）

作用：取得低分辨率图形方式屏幕上点（X，Y）的颜色值。其中（X，Y）为所取颜色值的点的坐标。一般X在0

～39之间，Y在0～47之间取值

4． PLOT

格式：PLOT X，Y

作用：在低分辨率图形方式屏幕的点（X，Y）处绘出一个色点，X及Y在0～39的范围内取值。坐标原点（0,0）在屏幕的左上角。点的颜色由COLOR语句设定。

5． HLIN

格式：HLIN X_1，X_2 AT Y

作用：在低辨率方式的屏幕上，绘出一条水平线。

（X_1，Y）为水平线的起点，（X_2，Y）为水平线的终点。

6． VLIN

格式：VLIN Y_1，Y_2 AT X

作用：在低分辨率图形方式屏幕上，绘出一条垂直线。

（X，Y_1）为垂直线的起点，（X，Y_2）为垂直线的终点。

例11.1：画出一个方框的程序。

```
10  GR：COLOR=9：HOME
20  HLIN  5，30  AT  5
30  VLIN  5，20  AT  30
40  HLIN  30， 5  AT  20
50  VLIN  20， 5  AT  5
60  FOR I=0  TO  300:NEXT I
70  PLOT 17，12
80  GOTO 10
```

7． TEXT

格式：TEXT

作用：从低分辨图形方式或高分辨图形方式转到满屏文本显示方式。

例11.2：在分区数N≤5 情况下绘制直方图的通用程序。

```
10  GR: COLOR=5
20  INPUT "请输入分区数N"; N
30  DIM D(N)
40  S=0:FOR I=1 TO N
50  READ D(I):S=S+D(I)
60  NEXT I
70  K=INT(28/N)
80  HOME
90  FOR I=1 TO N
100 FOR J=8+(I-1)*K TO 7+I*K
110 IF N>8 THEN 130
120 VLIN 39-D(I)*60/S, 39 AT J:GOTO 140
130 VLIN 39-D(I)*80/S, 39 AT J
140 NEXT J
150 NEXT I
160 END
170  DATA 30, 13, 24·5, 9, 11·6
```

11.2 BASIC高分辨率绘图语句

1. HGR

格式：HGR

作用：将屏幕（显示区首页）设定为高分辨率图形与文

本混合方式（HGR）。屏幕上部为280×160的图形区域，下部留4行作显示程序正文之用。

若要由高分辨图形与文本混合方式转到全屏纯高分辨图象显示（HGR2）状态，则用指令：POKE－16302，0

若要由HGR2状态返回到HGR状态，则用指令：POKE－16301，0

执行HGR命令时，将屏幕图形区清为黑色。对原有图形清除。

2．HGR2

格式：HGR2

作用：置屏幕第二页为高分辨图形状态，使屏幕变成280×192的图形区域（不留底部4行给程序正文用）并清除全屏（即全屏显黑色）。

3．HCOLOR＝n

格式：HCOLOR＝n

作用：供在高分辨图形方式下选择8种颜色。

n值所对应的8种颜色如下：

n值	颜色
0	黑
1	绿
2	蓝
3	白
4	黑2
5	红
6	黄
7	白2

4．HPLOT

格式1：HPLOT X，Y

格式2：HPLOT TO X_2，Y_2

格式3：HPLOT X_1，Y_1 TO X_2，Y_2〔{TOX_i，Y_i}〕

$i=3，4，5\cdots$

作用：

格式1　在坐标（X，Y）处画出一个色点。

格式2　表示以上一次新画的最后一个点到（X_2，Y_2）坐标之间画一条线。

格式3表示从（X_1，Y_1）到（X_2，Y_2）之间画一条线，再从（X_2，Y_2）到（X_3，Y_3）画一条线，依次画下去，其范围受屏幕限制，一条指令不得超过239个字符的限制。X和Y以及X_i和Y_i都可以由算术表达式来代替。

X的范围：0～279；Y的范围：0～159（HGR状态）或0～191（HGR2状态）。原点（0，0）在屏幕的左上角。

例如：下面的语句可画出三段相互连接的线段：

```
HPLOT 40，50  TO 80，90  TO 125、145  TO
40，50
```

例11.3：画一艘"飞船"的程序。

```
10  REM画一艘"飞船"的程序
20  HOME：HGR：HCOLOR＝5
30  HPLOT 120，120 TO 120，50 TO 140,20 TO
160，150 TO 160，120
40  HPLOT TO 180，170 TO 120，120
50  HPLOT 120，120 TO 120,170 TO 160,170 TO
160，120
```

```
60  END
```

例11.4：画花纹程序

```
3   REM这是一个画花纹的程序。
6   HGR2:A=5
10  X1=RND(1)*279:Y1=RND(1)*191
20  C=C+1:IF  C=8 THEN C=0
30  HCOLOR=C:Y=191
40  FOR X=0  TO 279 STEP A
50  HPLOT X, 191 TO X1, Y1 TO 279 -X, 0
60  NEXT X
70  FOR Y=0  TO 191 STEP A
80  HPLOT 279, 191-Y TO X1, Y1 TO 0, Y
90  NEXT Y
100 FOR I=1 TO 500:NEXT I
110  IF PEEK(-16384)=155 THEN  POKE-
16368, 0
115  TEXT
120  GOTO 10
130  END
```

说明：第10行随意选定一个图形中心。这个程序连续画出许多个互相重叠的图案，每画一个改变一次颜色（20，30两行）。第100行用于延时，使程序画完一个图案时停一停，以便使用者能看清楚图形。第110行跳出程序，按ESC键程序结束。

本程序画出的是漂亮的图案。

11.3 高分辨率图形的向量作图法

在高分辨率作图方式下还有另一种画图方法称作向量作图法。其原理是：将需要绘制的图形用一组“描绘向量”将其轨迹描绘出来，并将这组向量存放在内存的某个区域。这组“描绘向量”就构成内存中的“一张”图形定义表。然后用特定的命令调用此图形定义表中的“描绘向量”，图形就可以展现在屏幕上。

存放“描绘向量”的内存区中，每个字节被分成三段。如下图所示：

C		B			A		
7	6	5	4	3	2	1	0
D	D	P	D	D	P	D	D

如图：A、B两段中三位数的低二位及C段中的二位数称做D位；高位称P位。对D和P规定如下：

$$P=\begin{cases}0 & \text{不画点}\\ 1 & \text{画点}\end{cases}$$

$$DD=\begin{cases}00 & \uparrow\\ 01 & \rightarrow\\ 10 & \downarrow\\ 11 & \leftarrow\end{cases}$$

某段的P=1时，用此段作图时就先画一个点，然后依D位的数移动一个坐标点（按高分辨作图时，屏幕上坐标点共

可有192×280）；P＝0或无P位（C段）时，则根据D位的数，仅移动一下坐标点。例如，要画出下图11.1中的6个点，可以用下图11.2中表示的a～g 7个步骤来完成：



图11.1

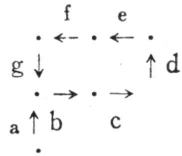


图11.2

其中a表示画一个点，然后上移一个坐标；b表示画一个点，然后右移一个坐标；c与b相同。但d则表示不画点，仅上移一个坐标。因此，a～g对应的描绘向量段应分别为

a：100

b：101

c：101

d：000（或00）

e：111

f：111

g：110

把各个段组成“描绘向量”时还有如下的规定：

①画图时，顺序从图形定义表中逐个取出“描绘向量”由A～C顺序作图。

②对每个（字节）“描绘向量”来说，如果C＝00，则C段不起作用；如果B也全为0，则只有A段起作用。

③不能是A、B、C三段全为0的“描绘向量。”三段全为0的向量（字节）用来表示图形定义表的结尾。

因此，上图的“描绘向量”应为下表所示，共占4个字节。

	C段		B段			A段		
向量 1	0	0	1	0	1	1	0	0
向量 2	0	0	0	0	0	1	0	1
向量 3	0	0	1	1	1	1	1	1
向量 4	0	0	0	0	0	1	1	0

内存中可以存放多个图形定义表，每个表中最后的一个单元为全0表示结尾，这些表的总数及存放位置由另一个表——索引表表示。下图是一安排有n个图形定义表的索引表。

S	图形定义表总数n	$(0 \leqslant n \leqslant 255)$
S+1	无用	
S+2	D1L	1号图形定义表首址D1
S+3	D1H	
S+4	D2L	2号图形定义表首址D2
S+5	D2H	
⋮	…	
S+2n	DnL	n号图形定义表首址Dn
S+2n+1	DnH	

图中S是索引表的首地址，表中主要存放了各个图形定义表中第一向量存放单元相对于S的相对地址。每个图形定义表，从这里开始存放“描绘向量”字节。最后存放一个全0字节。

如此，根据索引表便可逐次地找到各个图形定义表。并按规定逐段作图，直到遇到一个全0字节为止。如此作好n个表，就是所需要的图形。

使用此向量作图时，先要把索引表的首址（S）存放到$E8和$E9单元（$E8存其低位，$E9存放其高位。）

BASIC提供下述四个命令，使用这些命令在图形定义的基础上可画出各种图形（包括放大或缩小的图形，旋转后的图形等等）。

1. DRAW

格式：DRAW n AT X, Y

DRAW n

作用：根据图形定义表在屏幕上进行作图。

本命令根据第n个图形定义表作图。（X, Y）为图形的起点坐标。其中：$0 \leqslant n \leqslant 255$，$0 \leqslant X \leqslant 279$ 且 $0 \leqslant Y \leqslant 191$。使用第二种格式时，起点为上次使用DRAW、HDLOT或XDRAW时，所画图形的最后一点的坐标。若前面没有使用这些命令，则（0, 0）作为起点。

2. XDRAW

格式：XDRAWn AT X, Y

XDRAWn

作用：该命令格式及使用方法与DRAW一样，但作用相反，用于删除图形。如用DRAW画了一个图形，再用XDRAW画同一图形（命令参数全同）则清除已画的图

形。

3． SCALE

格式：SCALE＝n

作用：决定用DRAW或XDAW命令作图的比 例因子。o≤n≤255，当n＝1，一个向量段只移一个 坐标点；n＝2～255时，此移动量放大n倍；当n＝0时，相当于n＝256，图形放大256倍。

4． ROT

格式：ROT

作用：决定由DRAW或XDRAW命令作图时，图形的旋转度。0≤m≤255，当大于64时，按64取模，m值的含意与SCACE命令中的n值有关，即：

n＝
- 1
  - m＝0 右旋0度
  - m＝16 右旋90度
  - m＝32 右旋180度
  - m＝48 右旋270度
- 2　根据m可识别8种旋转度
- ≥5　根据m可识别64种旋转度

5． 高分辨率图形的向量作图实例：

例11.5

```
20 DATA 5，0，12，0，21，0，30，0，39，0，46，0
30 DATA 45，40，32，5，168，174，21，37，0
40 DATA 45，40，32，5，40，45，21，37，0
50 DATA 45，5，40，5，40，45，21，37，0
```

```
60  DATA 45, 45, 45, 45, 45, 37, 0
70  DATA 9,5,32, 5, 40, 168, 21, 21, 173,
37, 0
80  FOR I=768 TO 824
90  READ X:POKE I, X
100 NEXT I
110 POKE 232, 0:POKE 233, 3
120 HOME:HGR:HCOLOR=3:SCALE=1:ROT
=0
130 FOR X=10 TO 200 STEP 4
140 FOR I=1 TO 5
150 DRAW I AT X, 100
160 XDRAW I AT X, 100
170 NEXT I
180 NEXT X
190 GOTO 130
200 END
```

本程序运行时在屏幕上给出一个动态的运动图形。其索引表首址是在＄300（＝768），共有5个图形定义表。

DATA句中的每一个10进制数是由图形定义表中一个描绘向量（字节）的内容，按8位2进制数转换成10进制数的方法计算出来的。

例11.6 本程序可用选菜单方式，选择画出图形或画出十字框，程序中还运用了“定比例”SCALE及“旋转”ROT语句。

```
10  TEXT:CLEAR:POKE 216, 0:A=FRE(0)
30  GOTO 1000
```

```
50  REM选定颜色
52  C=INT(RND(1)*8)
53  IFC=OC THEN 52
54  HCOLOR=C
55  OC=C
56  RETURN
60  REM* 测验键盘
70  SW=0
74  IF PEEK(-16384)>127 THEN SW=1
76  POKE-16368, 0
78  RETURN
99  REM *填满屏幕
100 GOSUB 50
102 HPLOT 0, 0
103 CALL 62454
105 GOSUB 70
110 IF SW=1 THEN RETURN
115 GOTO 100
199 REM *绘图
200 ST=-ST:GOSUB 50
205 B=1:E=R
210 IF ST<1 THEN B=R:E=1
230 FOR I=B TO E STEP ST
235 ROT=I:SCALE=I
240 DRAW SH AT X, Y
242 HCOLOR=7-C
243 XDRAW SH AT X, Y
```

```
245  NEXT I
250  GOSUB 70
255  IF SW=1 THEN RETURN
260 CALL 62454:GOTO 200
999 REM 装入描绘向量表首地址
1000 POKE 232, 00:POKE 233, 03
1010 REM *装入图形定义表数据
1020 FOR LOC=768 TO 783
1030 READ BYTE:POKE LOC, BYTE
1040 NEXT LOC
1050 DATA 02, 00, 06, 00, 09, 00, 44, 62, 00,
     44, 46, 62, 62, 60, 44, 00
1060 ROT=0 :SCALE=8
1070 X=140
1080 ST=1
1999 REM *主程序
2000 TEXT:HOME:PRINT TAB(10)"**向量
作图实例**"
2001 PRINT TAB(18)"01/05/1988"
2002 PRINT "菜单:" :PRINT "………"
2004 PRINT TAB( 5 ) "1……方形旋转和大小变
化":PRINT
2006 PRINT TAB( 5 ) "2……十字框": PRINT
2008 PRINT TAB( 5 ) "3……闪烁翻腾的波浪"
     :PRINT
2010 PRINT TAB( 5 ) "0……退出"
2020 VTAB(14):CALL -958:INPUT"请选择";
```

```
    N:SH=N
2025 IF N=0 THEN VTAB 4:CALL-958:V
    TAB 12:PRINT "再见!"
2027 END
2030 IF N<1 OR N>3 THEN PRINT CHR$
    (7)
2035 GOTO 2020
2040 HGR:POKE-16302, 0
2050 CALL 62454
2060 IF SH=1 THEN Y=95
2065 R=80
2070 IF SH=2 THEN Y=110
2075 R=48
2080 ON N GOSUB 200, 200, 100
2090 GOTO 2000
```

本程序把所定义的图形（在1050行定义）显示在高分辨率屏幕上，当变量 I 的值增加时，使其图的大小变化并旋转。

语句功能简介如下：

语句号	功能
10	清屏幕和内存
50～59	选定新颜色
69～78	检测键盘
99～115	将屏幕用上一次颜色填满
199～260	定图形大小，绘出并旋转；定义负的增加值ST；选择颜色

	值；
	定义FOR…NEXT循环中的初值B及终止值E；按I值旋转图形；
	按I值定图形的大小；
	在X，Y处画出图形SH；
	检测键盘，GOSUB 70；
	用上一次颜色将屏幕填满。
1000～1050	存图形表入内存，用1050行的图形表数据。
1060	定ROT值，定SCALE值及X的位置
	设ST＝1
2000～2030	主程序段
2040	设定图形状态
2050	用上次颜色填满屏幕
2060～2070	设定绘图Y位置，定出绘图范围R值
2080	执行选择的子程序
2090	返回主程序

运行此程序时，“选择“1”，表演方形的旋转和大小变化。选择“2”，表演十字框。选择“3”，表演闪烁翻腾的“波浪”。选择“0”，程序运行结束。

11.4 BASIC音乐程序的编制

中华学习机内部有一个扬声器，在程序的控制下可以产生音响、演奏乐曲。本节介绍如何使用中华学习机发音功能

语句编制音乐程序。发音功能语句（MUSIC）的格式和作用如下：

格式：MUSIC X，Y

作用：定义一个音阶和音长，并使计算机中的扬声器发音。其中X表示发音的频率，Y表示发音的时间，X、Y的取值范围在0～255之间。它们的取值与对应的音阶和音长如下表：

X值	255	228	205	192	171	152	140	128	114	102
音阶	$\underset{\cdot}{5}$	$\underset{\cdot}{6}$	$\underset{\cdot}{7}$	1	2	3	4	5	6	7

X值	95	84	75	68	62
音阶	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$	$\dot{5}$

y值	30	70	110	160	255
音长	1/4拍	1/2拍	1拍	2拍	4拍

下面用例子来说明如何编制音乐程序。

例11.7　一个音阶旋律的程序可设计如下：

```
10  DATA 225, 160, 228, 160, 205, 160, 192,
    160, 171
20  DATA 160, 152, 160, 140, 160, 128, 160,
    114, 160, 102
30  DATA 160, 95, 160, 84, 160, 75, 160, 68,
    160, 62, 160
40  FOR I=1 TO 15
```

```
50  READ X, Y
60  MUSIC X, Y
70  NEXT I
80  END
```

当选好一首乐曲之后，用上面的方法先列出这首乐曲每一音符的两个数字（即音阶和节拍）编入DATA语句中。

例11.8　“欢乐舞曲”的程序

```
10  DIM A(35), B(35)
20  DATA 192, 110, 192, 70, 171, 70, 152,110,
    152, 70, 140, 70, 128, 110, 114, 70,128, 70,
    152, 160
30  DATA 128, 110, 140, 70, 152, 70, 171,160,
    140, 110, 152, 70, 171, 70, 192, 160
40  DATA 192, 110, 192, 70, 171, 70, 152,110,
    152, 70, 140, 70 , 128, 110, 114, 70, 128,
    70, 152, 160
50  DATA 128, 110, 140, 70, 152, 70,171,110,
    152, 70, 171, 70, 192, 230
60  FOR I=1 TO 35
70  READ A(I), B(I)
80  NEXT I
90  FOR J=1 TO 35
100 MUSIC A(J), B(J)
110  NEXT J
120 END
```

“欢乐舞曲”

1=C　4/4

1 12 3 34 | 5 65 3— | 5 43 2— | 4 32 1— |

1 12 3 34 | 5 65 3— | 5 43 2 32 | 1——0 |

例11.9 流行游戏音乐程序

```
10  DATA 192, 70, 171, 70, 152, 110, 128, 110,
    128, 135, 114, 70, 128, 110, 152, 110, 192,
    135,171,70, 152, 110, 152, 110
20  DATA 171, 110, 192, 110, 171, 230 , 192 ,
    70, 171, 70, 152, 110, 128, 110, 128, 135,
    114, 70, 128, 110, 152, 110, 192, 135, 171,
    70
30  DATA 152, 110, 152, 110, 171, 110 , 171,
    110, 192, 230
40  DATA 140, 160, 140, 160, 114, 110, 114 ,
    160, 114, 110, 128, 135, 128, 70, 152, 110,
    192, 110, 171, 230, 192, 70, 171, 70 , 152,
    110, 128, 110
50  DATA 128, 135, 114, 70, 128, 110, 152,110,
    192, 135, 171, 70, 152, 110, 152, 110 , 171,
    135, 171, 70, 192, 230
60  FOR I=1 TO 5
70  RESTORE
80  FOR J=1 TO 55
90  READ K, L
100 MUSIC K, L
110 NEXT J
120 NEXT I
```

130 END

流行游戏音乐：

1=C　2/4

12 | 3 5 | 5·6 | 5 3 | 1·2 | 3 3 | 2 1 | 2 — | 2

12 | 3 5 | 5·6 | 5 3 | 1·2 | 3 3 | 2 2 | 1 — | 1

0 | 4 — | 4 — | 6 6 | 6 6 | 5·5 | 3 1 | 2 — | 2

12 | 3 5 | 5·6 | 5 3 | 1·2 | 3 3 | 2·2 | 1 — | 1 ‖

习　题

1. 试编写一个低分辨率彩色作图程序。
2. 试编写一个高分辨率彩色作图程序。
3. 试用高分辨率图形的向量作图法编写一个绘图程序
4. 试编写二个音乐程序。

第十二章　操作系统概述

12.1　什么是操作系统

操作系统是一套为了提高计算机的工作能力，扩展计算机的功能而配的程序系统。中华学习机使用DOS 3.3磁盘操作系统，DOS是英文Disk Operatiny System（磁盘 操作系统）的缩写，3.3是该磁盘操作系统的版本号。

磁盘是计算机常用的外存贮器，可分为硬磁盘和软磁盘两种，简称硬盘和软盘。中华学习机只能使用软盘作外存贮器。软盘要放入磁盘驱动器中使用，磁盘驱动器也称为磁盘机。软盘在磁盘机中是高速旋转的，它通过移动的磁头存取信息。中华学习机的一个软盘在DOS 3.3系统下有143K字节的随机存取容量。

12.2　操作系统的装入

DOS 3.3 磁盘操作系统是以某种数据 形式存储在软盘片上的。如果想使用DOS 3.3 磁盘操作系 统 进行工作，首要的问题是如何将该操作系统程序从软盘上装入到主机内存中，然后才是怎样利用操作系统所提供的命令进行操作。这种操作系统装入内存，并使其工作的过程称为磁盘操作系统的引导。

在中华学习机上可以有两种方式来进行磁盘操作系统的引导—开机引导和命令方式引导。

12.2.1 开机引导

中华学习机的固化软件中有一个自启动监控程序，开机后机器首先执行的是这段程序。该程序从连机的软盘驱动器（设备号为：S_6，D_1）中读磁盘，装入磁盘操作系统。因此，只要用户在开机前将 DOS 3.3磁盘操作系统软盘插入软盘驱动器，并关好驱动器的门。然后，再打开主机电源开关，机器便会自动将软盘中的操作系统装入内存。

这种开机引导磁盘操作系统的方式称为“冷启动”方式或“自启动”方式。

12.2.2 命令方式引导

如果主机已加电运行，需要装入或重新装入磁盘操作系统，可以将操作系统盘插入软盘驱动器，根据下列所述各种状态，打入相应命令，即可进行操作系统的引导。

系统状态	提示符	打入命令
BASIC	]	PR #6或IN #6
整数BASIC	>	PR $6或IN#6
监控程序	*	6CTRL-P或C600G
小汇编	!	$C600G

这种在主机已加电情况下，引导操作系统的方法又称为“热启动”。

12.3 操作系统的主要应用

下面介绍的操作系统命令，它们均可以在 BASIC 状态“]”下直接使用，就如同对话方式下的 BASIC 语句用法

一样。

12.3.1 新盘片的初始化

新购买的空白软盘片，上面没有记录任何信息，在对这样的盘片用于记录程序、数据等信息资料之前，必须对其进行格式化。格式化的作用是在磁盘上按磁道和扇区来划分地址，并把地址以数据的形式记录在软盘上，便于读/写数据，按其地址进行查找。

在DOS 3.3 磁盘操作系统命令中，提供了一条初始化盘片的命令（INIT）。利用此命令，可以完成对软盘进行格式的操作。

初始化操作的步骤：

1. 用DOS 3.3磁盘操作系统进行启动。

2. 打入NEW命令并从盘上装入一个BASIC程序。如例12.1：

```
10 REM GREETING PROGRAM
20 TEXT:HOME
30 PRINT "DOS3.3 DISK CREATED ON CEC
   -I"
40 PRINT "BY X. Y. WANG, 27-MAY-1988"
50 PRINT "FOR DOS BASIC PROGRAM STO
   RAGE"
60 END
```

读者也可以输入别的程序，试运行一次认为满意后继续下一步操作。

3. 取出系统盘，插入空白盘，并关上驱动器门。

4. 打入命令：

INIT　<文件名>↵

当屏幕上再次出现提示光标后（约30秒钟）表示初始化工作完毕。第4步中命令INIT 后面的文件名，由用户任给（如：HELLO）。

所有经过初始化操作的空白盘片中都存有 DOS 磁盘操作系统程序，也都可作为操作系统的引导盘进行启动。

若我们不取出新初始化的盘片，关掉主机电源，数秒钟后，再重新开主机电源开关，屏幕上将出现如下信息：

DOS 3.3 DISK CREATED ON CEC-I

BY X. Y. WANG, 27—MAY—1988

FOR DOS BASIC PROGRAM STORAGE

]

这表示用新初始化的盘片引导DOS 3.3操作系统成功。

12.3.2　列文件目录

命令格式：CATALOG

作用：显示软盘中文件的目录。

如果我们在软盘驱动器中插入在12.3.1节所初始化的新盘，并打入命令：

] CATALOG↵

屏幕上将出现如下信息：

DISK VOLUME 254

A 002 HELLO

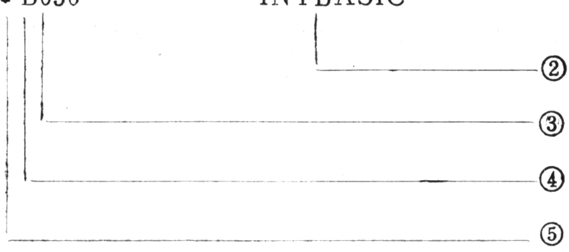
如果换上DOS 3.3系统主盘，再打入 CATALOG 命令，屏幕上将显示如下信息：

DISK　　VOLUME 254————①

*A008　　HELLO

*I018	ANIMALS
T003	APPLE PROMS
*I006	APPLE SOFT
*I026	APPLE V ISION
*I017	BIORH Y THM
*B010	B00 T13
*A006	BRIAN' S THEME
*B003	CHAIN
*I009	COLOR DEMO
*A009	COLOR DEMOSOFT
*I009	COPY
*B003	COPY, OBTO
*A009	COPYA
*A010	EXECDEMO
*B020	FID
*B050	FPBASIC

* B050　　INTBASIC



说明：

①该盘片所占有的卷号。一张盘片只能有一个卷号，它是在初始化命令执行过程中所确定的。

②文件名，每个文件名最大长度为30个字符。

③文件在软盘上所占用的扇区数。

④文件类型：

A　表示APPLESOFT程序文件。

I　表示整型BASIC程序文件。

B　表示2进制代码文件。

T　表示文本文件。

R　表示由EDASM汇编程序所产生的可浮动目标文件。

L　表示由LISA汇编程序所产生的文件。

⑤　文件保护标志。它是操作系统所提供的对文件进行写保护的一项措施。

如果该文件有星号(*)标志，则表示DOS对该文件进行了封锁保护，它只允许用户对该文件执行读操作，而不能对其执行改名、删除或写操作。否则，系统会给出：FILE LOCKED信息。

在列目录过程中，可以用CTRL—S键暂停列目录，再按任意键使其继续列目录。如果目录内容多于屏幕一次能够显示的条目时，可按任意键使其继续显示。

12.3.3　保存BASIC程序

若我们在主机上编制了一个BASIC语言程序，需要保存起来，随时把这个程序调入主机修改、调试、运行等。则我们可以使用DOS操作系统提供的存盘命令，满足要求。

命令格式：SAVE　<文件名>

作用：用给它的文件名将当前内存中的BASIC程序保存到软盘上。

其文件名是以字母打头的字符序列(但不能含有逗号)，

文件名的最大长度为30个字符。

例12.2 我们在内存键入了一个BASIC程序：

```
  5 FOR I=1 TO 10
 10 A=INT(RND(I)*1000)
 20 INPUT "输入一个您猜的数"; B
 30 S=S+1
 40 IF A=B THEN 80
 50 IF A<B THEN PRINT "大了"
 60 IF A>B THEN PRINT "小了"
 70 GOTO 20
 80 PRINT "猜数正确"
 90 PRINT "您猜数的次数", S
 92 S=0
 95 NEXT I
100 END
```

用RUN命令运行该程序，试用正确后，可以用SAVE命令将该程序保存到工作盘上，命令用法如下：

] SAVE GUESS NUMBER↲

该程序就以GUESS NUMBER名字保存在键盘上。以后就可以多次使用这段程序。

用列文件目录命令，即可看到此文件名。

12.3.4 把BASIC程序调入内存

如果我们希望从软盘装入一个BASIC程序到内存。以载入GUESS NUMBER程序为例，则键入的命令是：

] LOAD GUESS NUMBER↲

用LOAD命令把程序装入内存时，清除主机内原有的

BASIC程序，由新装入的程序替换原有的程序而软盘上的程序不受影响。

如LOAD命令中指定的文件名的程序不在当前软盘上，将显示：FILE NOT FOUND（文件没有找到）。

12.3.5 复制磁盘文件

为了防止软盘中的信息在使用中被破坏，通常是采用复制副本的方法。

COPYA和COPY程序是DOS操作系统提供的两个软盘复制程序。

在软盘复制过程中，被复制的盘片称为“源盘”，而复制后的盘片称为“复制盘”。复制的目的是把“源盘”上的全部信息，一道一道地复写到“复制盘”上。复制盘是源盘的一个副本。

1. 全盘复制，其复制过程是：

①将DOS 3.3 系统主盘插入软盘驱动器，键入命令：

RUN COPYA （或RUN COPY）

②屏幕显示如下：

APPLE DISKETTE DUPLICATION PROG-AM

ORIGINAL SLOT:DEFAULT=

此时，依次回答源盘所在驱动器的槽口号和设备号分别为6和1。

③屏幕显示如下：

DUPLICATION SLOT:DEFAUAT=

系统询问复制盘所在驱动器的槽口号和设备号，对于单软盘驱动器来说，仍然依次回答6和1。（如果在多盘驱动

系统机器上工作，则回答复制盘所在的驱动器的槽口号和设备号）。

④屏幕显示如下：

APPLE DISKETTE DUPLICATION PROGRAM

ORIGINAL　　SLOT：6
　　　　　　DRIVE：1
DUPLICATE　SLOT：6
　　　　　　DRIVE：1

–PRESS 'RETURN' KEY TO BEGIN COPY—

⑤按RETURN键，开始进入复制工作。

在单驱动器情况下，程序将提示用户何时抽入源盘，何时插入复制盘。

⑥盘片复制完毕后，屏幕提示：

DO YOU WISH TO MAKE ANOTHER COPY?

意思是问用户是否还复制别的软盘，如果需要则键入Y键，否则键入N键退出复制程序。

在使用复制命令时注意：

①在盘片的复制过程中，如没有适时插入源盘；或驱动器门没有关好，或源盘是加密的，或源盘上有的文件信息已被损坏，则可能出现信息：

***** UNABLE TO READ *****

意思是不可读。

②如果没有适时插入复制盘，或复制盘上贴有写保护，或指定的驱动器槽口号和设备号有错，出现的信息是：

******UNABLE TO WRITE******

意思是不可写。

③在复制盘片操作之前，对源盘片贴上写保护，可以防止误操作可能引起的损坏源盘片问题。

2． 单个文件的复制。

在 DOS 3.3 磁盘操作系统中有一个用于磁盘文件管理的实用程序FID，以菜单选择的方式对指定的磁盘文件进行有关操作。它的第一个功能，可实现单个文件的复制。

FID的执行命令：BRUN FID↵

屏幕显示如下：

```
* * * * * * * * * * * * * * * * * * * * * * *
*     APPLE Ⅱ  FILE  DEVELOPER              *
*                                           *
*           FID  VERSIONM                   *
*                                           *
* COPYRIGHT 1979 APPLE COMP U TER           *
* INC                                       *
* * * * * * * * * * * * * * * * * * * * * * *
```

CHSOOE ONE THE FOLLOWING OPTIONS

（选择如下操作中的一种）

（1）COPY FILES　　复制文件

（2）CATALOG　　列文件目录

（3）SPACE ON DISK　　磁盘空间

（4）UNLOCK FILES　　解锁

（5）LOCK FILES　　加锁

（6）DELETE FILES　　删除文件

（7）RESET SLOT DRIVE重置槽口和驱动器

(8) VERIFY FILES　　　校验文件

(9) QUIT　　　　　　　退出

WHICH WOULD YOU LIKE?

上述菜单中1—9为各种不同的操作，键入相应的数字及回连键选择所需要的操作。

COPY FILES 将源盘上被指定的文件复制到复制盘上。复制盘必须是事先经过初始化操作的软盘。

除功能9是退出FID程序执行外，其它功能实现对指定文件执行相应操作。

12.3.6 删除文件

对于软盘上存储的已经确认没有保留价值的文件。如：调试程序过程中留下的不用的旧版本程序。旧的数据文件，都可用删除命令把它们删除掉。

命令格式：DELETE <文件名>↵

作用：删除磁盘中指定文件名的文件。

假如我们需要删除盘上的 AB 文件。应键入如下的命令：

DELETE AB↵

本命令键入后，软盘上的AB文件将从文件目录中删去，不可再恢复该文件。因此，一定要确认无保留价值的文件才能用本删除命令删除掉。为了防止一时不慎删掉，有用的文件，系统提供有LOCK命令对文件加写封锁标志，用LOCK命令加了这种标志的文件，DELETE命令不能删除，并显示：FILE LOCKED。

如果软盘贴有写保护，将得信息：

WRITE PROTECED

意思是软盘具有写保护。删除命令不能执行。

如果在 DELETE 命令中指定的文件名不在软盘上，将显示：FILE NOT FOUND

意思是指定的文件在软盘上没找到。

12.4 操作系统的主要命令及其作用

一条 DOS 命令可以带有若干命令参数，参数之间必须用逗号隔开。

DOS 命令的一般格式如下：

```
COMMAND       f [,Ss] [,Dd] [,Vv]
   |          └────────┬────────┘
命令保留字          命令参数
```

格式中方括号之中的项为可选择的命令参数，大写字母为参数标记，小写字母为参数。可选择命令参数顺序是任意的。

命令参数说明：

f 文件名，以字母打头的字符串组成（不能用逗号），不超过30个ASCII字符。

Ss 设定软盘驱动器所在的槽口号，S 值可以是 1 — 7 之间的整数（但不能为 3 ）。

中华学习机的盘驱接口上所接盘驱的槽口号为 S＝ 6 。

Dd 设定软盘驱动器在盘驱接口卡上的设备号。d值必须是 1 或 2 。

中华学习机盘驱接口上所接盘驱的设备号d＝1

Vv 软盘片的卷号，卷号值V是 1—254之间的整数，1 张软盘只用一个卷号。

卷号在用 INIT 命令进行软盘初始化操作时给定，不可

更改，它起核对盘片的作用。

如果省略了命令参数，则被认为是与上次使用的相同。

12.4.1 管理磁盘文件的命令

1. CATALOG

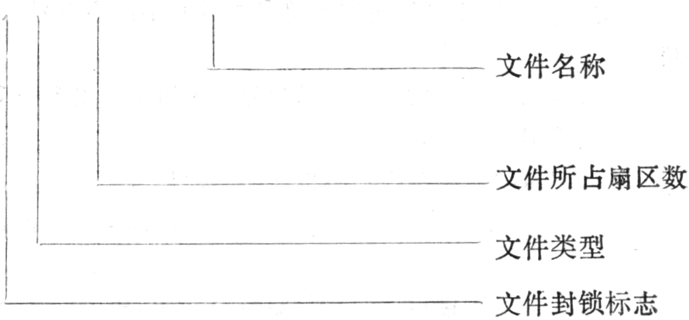
用途：列磁盘文件目录。

格式：CATALOG [,Ss] [,Dd] [,Vv]

说明：该命令将按命令指定的软盘驱动器列出该盘驱上盘片的文件目录。目录中包括：磁盘的名称，文件所占用的盘片的扇区数，文件的类型和文件的封锁标志等等。

例如：

```
] CATALOG↲
DISK VOLUME 254————磁盘卷号
*A008 HELLO
*I018  ANIMALS
 T003 APPLEPROMS
```



2. LOCK

用途：置文件封锁。

格式：LOCK f [, Ss] [, Dd] [, Vv]

说明：该命令对指定的文件置为封锁（写保护）状态，使得该文件不会因意外的写操作而遭破坏。

对于被封锁的文件，用户只能执行 LOAD/READ-EXEC/VERIFY 操作，而不能对其执行 SAVE/WRITE/RENAME/DELETE等操作。否则将会显示：FILE LOCKED。意思是文件被封锁。用 LOCK 命令封锁的文件，列磁盘文件目录时，该文件类型标志的左边将有一星号标志（*）。

例如，插入刚刚初始化完毕的工作盘，列出磁盘文件目录

```
] CATALOG
DISK VOLUME 010
A002  HELLO
```

此时执行LOCK HELLO 命令，再列文件目录，HELLO文件已被封锁。

```
] LOCK HELLO↵
] CATALOG↵
DISK VOLUME 010
*A002 HELLO
```

3. UNLOCK

用途：解除文件封锁。

格式：UNLOCK f [, Ss] [, Dd] [, Vv]

说明：该命令解除对文件的封锁。用该命令解除文件封锁时，取消其封锁标志，并可进行删除、重新命名，修改或重写等操作。

LOCK和 UNLOCK 命令可对任意类型的文件进行操

作。

4. RENAME

用途：修改文件名。

格式：RENAME f_1，f_2［，Ss］［，Dd］［，Vv］

说明：该命令对指定的文件重新命名。用文件名f_2取代文件名f_1，可以对任何类型的文件执行该操作。改名后的文件其内容不受任何影响。

f_1是软盘上原有的文件名，f_2必须是软盘上没有的新文件名，f_1文件必须没被封锁。

例：若盘上目录是：

```
]CATALOG↵
DISK VOLUME 010
*A 002 HELLO
 A 009 OLDNAME
```

将OLDNAME文件改名为 NEW NAME，其命令如下：

```
]RENAME OLDNAME, NEWNAME↵
]CATALOG↵
DISK VOLUME 010
*A002 HELLO
 A009 NEWNAME
```

注意　RENAME命令不检查文件名f_2是否在软盘上已经存在，因此有可能由此命令造成在一张盘上多个同名文件的混乱。

5. DELETE

用途：删除文件。

格式：DELETE f［，Ss］［，Dd］［，Vv］

说明：该命令删除软盘上指定的文件，文件是未被封锁的任意类型文件。

如果指定删除的文件不存在，则出现信息：FILE NOT FOUND。如果指定删除的文件是被封锁的文件，则出现信息：FILE LOCKED。

6. VERIFY

用途：文件校验。

格式：VERIFY f [, Ss] [, Dd] [, Vv]

说明：该命令将通过读操作对指定文件f的校验和进行检查，以确定该文件在软盘上的信息是否正确。

12.4.2 **与BASIC程序有关的命令**

1. SAVE

用途：保存BASIC程序

格式：SAVE f [, Ss] [, Dd] [, Vv]

说明：该命令把当前主机内存中的BASIC程序保存到软盘上。内存中的文件不变。

f是所指定软盘副本的文件名。文件的类型根据当前系统所处BASIC语言状态确定。如果盘上没有该文件名，则在软盘上建立该文件。如果该文件名在软盘上已存在，并且文件类型与系统的当前状态相吻合，则用当前内存中的BASIC程序替代原有文件。如果软盘上的文件类型和主存中BASIC程序类型不一致，则显示：FILE TYPE MISMATCH。

2. LOAD

用途：装入BASIC程序。

格式：LOAD f [, Ss] [, Dd] [, Vv]

说明：该命令从软盘上装入一个类型为A或I的 BASIC

程序。LOAD命令装入BASIC程序之前，首先检查磁盘文件类型，根据装入的是A类型文件还是I类型文件，转入APPLESOFT解释程序或整数BASIC解释程序，然后才将该文件的内容装入到主机内存中。

在装入BASIC程序时，软盘上的文件保持不变，原先内存中的程序被清除。

如果没有用DOS 3.3系统主盘进行启动，或没有将整数BASIC解释程序装入到主机内存中，而要装入的却是I类文件，将会显示：LANGUAGE NOT AVAILABEL。如果欲装入的文件的类型不是A或I，则显示：FILE TYPE MISMATCH。如果指定装入的文件软盘不存在，则显示：FILE NOT FOUND。

3. RUN

用途：运行BASIC程序

格式：RUN f [, Ss] [, Dd] [, Vv]

说明：该命令将软盘上名为f的BASIC程序装入到内存并执行之。其执行步骤是：①执行DOS的LOAD命令。②执行BASIC的RUN命令。

如果命令中无f，即不指定文件名，则运行内存中当前现行的BASIC程序。如果RUN后跟的是行号而不是文件名，则从指定行号开始运行内存中的BASIC程序。

12.4.3 用于语言转换的命令

1. INT

功能：进入整数BASIC。

格式：INT

说明：该命令使系统进入整数BASIC状态，并清除内

存中现有的程序，恢复有关程序指针。

该命令不带任何参数，不能指定软盘驱动器的槽号和设备号。如果机内没有装入整数 BASIC 解释程序，系统提示如下信息：

LANGUAGE NOT ANAILABEL。

2. FP

功能：进入浮点BASIC。

格式：FP [，Ss] [，Dd]

说明：该命令使系统进入浮点 BASIC 状态。并清除内存中现有的BASIC程序，恢复有关程序指针。

如果我们因运行了某个 BASIC 程序，使系统有关指针改变，欲装入第二个程序执行时，系统回答："OUT OF MEMORY"信息拒绝执行时，可以使用 FP 命令。如果用户希望清除当前内存中的BASIC程序，也可以使用FP命令。从整数BASIC程序状态回到浮点BASIC状态，也可以使用FP命令

习　题

1. 上机初始化一个新盘片。

2. 上机复制磁盘及指定文件。

3. 试编写一个简单的BASIC程序，并练习CATALOG、LOCK、UNLOCK、RENAME、SAVE、LOAD RUN、DELETE等命令。

第十三章　中华学习机的汉字系统

13.1　汉字系统的功能

中华学习机具有较强的汉字处理能力。其汉字功能部件由国标一、二级全点阵汉字字库（二片1兆位ROM）和汉字处理程序（一片27256 ROM）构成，它占用系统的3号槽口。

中华学习机汉字系统向用户提供拼音、区位二种汉字输入方案，尚为用户保留有两个供扩充其它汉字输入方案的功能键F_4，F_5。每屏显示17×10个汉字，34×10个ASCII字符。汉字打印可提供15种字型。在BASIC程序中，汉字的内码占用3个字节：由汉字引导符7F加上区码、位码构成，ASCII字符则占用一个字节。BASIC中的各个语句均可在汉字状态下使用（除FLASH、PR#、IN#之外）。

13.2　汉字系统的进入与退出

中华学习机的汉字功能，可以通过软件命令或键盘上的命令键方便地进入或退出汉字系统。

13.2.1　汉字系统的进入

1.直接键入“中文”键，适用于BASIC、监控或DOS操作系统三种方式中。

2.在BASIC程序或DOS操作系统状态下，还可用PR#3

命令进入（在程序方式下，该命令必须尾随一个PRINT语句）。

3.在监控方式下，可使用命令3CTRL-P或3CTRL-K进入

说明：当进入汉字系统时，屏幕显示为高分辨图形的第2页，内存的程序不受影响，HIMEM：指针指向内存地址$9200

13.2.2 汉字系统的退出

1.在BASIC、监控或DOS操作系统三种方式下，都适于直接用键入“西文”键退出。

2.在BASIC程序中，可以执行命令TEXT或PRINT CHR$(17)退出。

3.在监控方式下，还可以使用命令C33AG退出。

退出汉字系统后，屏幕显示为文本的第1页，内存的程序不受影响，HIMEM：指针仍指向内存地址$9200。

13.3 汉字的输入方法

13.3.1 输入法的选择

进入汉字系统后，系统提供如下的选择功能键，并按用户的选择提供相应的汉字输入方法。

F1或CTRL-A——字母方式

F2或CTRL-L——拼音输入方式

F3或CTRL-W——区位码输入方式

F4或CTRL-T——保留供扩充新的输入方案

F5或CTRL-F——保留供扩充新的输入方案

CTRL-O键控制状态提示符显示或不显示。

13.3.2 拼音输入法

在汉字 系统状态下，键入“F2”键（或CTRL-L），屏幕提示行（在显示屏的下面）显示“拼音：”字样，表示进入了拼音输入方式。

中华学习机的拼音输入方式是全拼音输入方式，即按声母、韵母依次键入，系统根据输入的音节在状态提示行上显示出该音节的所有汉字，包括不同声调的字。

使用26个字母键作为输入码，查找一个汉字最多键入6个字母，大小写字母均可。当敲入汉字拼音的第一个字母后屏幕上提示出以该字母打头的音节中最常用的6个汉字供选择。若有需要输入的汉字，直接键入该汉字的编号对应的数字键即可；若没有，则继续依次键入后面的字母。当键入所有字母（1～6个）后，屏幕提示行显示出6个同音节 汉 字，若显示中没有所需的汉字，则用“>”键，查找后面的同音汉字，直到计算机发出“嘟”声，表示该音节同音汉字全部提示完毕。

若要返回查看前面的汉字，则键入“<”键，即显示前一幕汉字的6个汉字，如此可返回到该音节同音汉字中的第一幕汉字。

在向前或向后查找中，一旦提示行中出现了用户需要的汉字，就键入此汉字前面的序号，该汉字就会显示在屏幕上光标的位置，且汉字内码同时输入到键盘输入缓冲区中。如若此时连续键入相应的序号或向前向后查找的“<”或“>”键，可以连续地输入有关汉字。

在输入过程中，若拼音码敲错了，可按“◁”键，使状

态提示行上最后敲错的拼音字母被删除，或按空格键，可删除已输入到状态行的所有拼音字母。此时，再按“◁”键，将删除已键入到内存中的最后一个汉字。

13·3·3 区位码输入法

在汉字系统状态下，按“F3”键（或CTRL-W）后，屏幕状态提示行出现“区位：”字样，表示当前是区位码输入方式。

区位码输入方式使用国家标准GB2381-80区位方式输入汉字。此种输入方式，仅使用0～9共10个数字键输入汉字。国家标准中的每个汉字和图形字符均用4位数字键输入。

例如，国标中规定54区48位置上的汉字是“中”字（请参见本书的附录Ⅲ），在区位方式下，键入5448后，“中”字就输入到计算机中，并显示在屏幕的光标位置。

区位码　　汉字

54　48　**中**

⌣　⌣

区号　位号

国家标准中规定，00～10区为图形字符，16～55区为国家标准一级汉字，56～87区为国家标准二级汉字。每个区有94个汉字或字符，其位号为01～94。

在区位码输入汉字过程中，如果有一输入码按错，可用“◁”键来删除。如在输入作为区位码的数字键后，又输入一个数字的字符键，则系统将接收该数字符键，同时，可继续输入区位码。如果键入的区位码不在国标码所规定的范围内，系统会发出蜂鸣警告用户，拒绝接收这样的区位码，等待用户重新键入或删除修改。

13.3.4 特殊符号的输入法

在选择“F2”进入拼音方式后，键入－（减号）、＝（等号）、＼（斜线）可以选择输入标点符号、算术运算符号和制表线等。这些特殊符号说明如下：

按－（减号）可输入的符号有：

，。、；？！：“” ‘’ …

—●〔〈〕〉《「『》」』

〖【｛〗】｝

按＝（等号）可输入的符号有：

￥％＋－×÷±＝≈

＞＜≥≤／（［）］Σ～

①②③④⑤⑥⑦⑧⑨⑩

按＼（斜线）可输入的符号有：

－｜└┘┌┐├┤┴┬┼

13.4 汉字的打印输出

在汉字系统中，汉字的打印驱动程序是针对MX-80Ⅲ型打印机设计的，它要求打印机的接口为Centronics接口，打印机接口卡必须定义为I/O槽口的1号槽。

进入汉字系统后，可以使用以下的BASIC命令来控制和打印汉字。

13.4.1 设置打印汉字的字型号

格式：POKE 1659，N

说明：N＝0为不打印，进入汉字系统时，打印方式自动置为0。

N＝1～15，按系统定义的15种字型进行打印。

13.4.2 设置字间距

格式：POKE　1787，N

说明：N为字间距。N的取值范围为0～255，即字间距可在0至255个点之间，其系统设置为1。

13.4.3 设置行距

格式：POKE　1915，N

说明：N为行间距，取值范围为0～255，即0至255个点

13.4.4 设置行允许字数

格式：POKE　2043，N

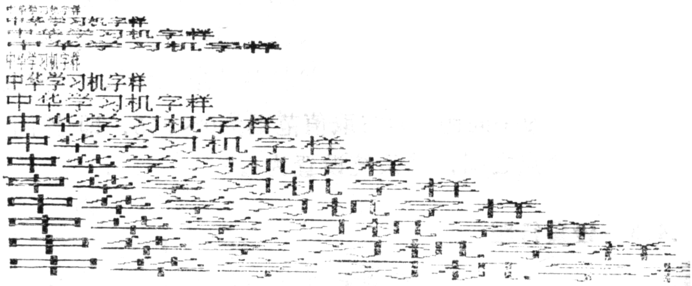
说明：N为一行所允许打印的汉字个数。取值为0至255一个汉字占两个字符的位置。其系统设置为40。

一行允许打印的数字与打印机本身的缓冲区大小有关，一般80列打印机的缓存为400个字节左右，因此，一行允许打印的汉字个数与所选择的打印字型及设定的字间距有关。

13.4.5 汉字打印实例程序

```
]LIST↵
10 FOR N=1TO15：POKE 1659,N：PRINT "中华学习机字样"：NEXT N
]RUN↵
```

从本例可见，打印的字体根据设置的字号而定，若设置的字号大于15，则系统以16为模。若设置的字号为0，系统退出汉字打印模式。



13.4.6　对某些打印机适应汉字打印的修改

在汉字管理程序中，打印机驱动程序仅适用于九针图形打印机。其型号要求必须与EPSONMX-80Ⅲ型打印机相兼容，如：FX-80Ⅲ、RX-80Ⅲ、CP-80Ⅲ、FX-100⁺和YAMATO等型号的打印机。而对于MX-80Ⅱ、FX-80Ⅱ、RX-80Ⅱ等型号的打印机均不能直接使用。

为使上述Ⅱ型打印机能够连接中华学习机打印汉字，需要修改打印控制命令，

1.修改走纸命令

因MX-80Ⅱ型打印机，没有n/216英寸走纸功能，所以必须把打印驱动程序中使用的ESC3m命令修改成ESCAM命令，才能打印汉字。

这个命令序列在打印驱动程序中分别放在LFCODEO LFCODE2、LFCODE4（控制上半行打印走纸）和LF-CODEO、LFCODE2、LFCODE3（控制下半行打印走纸）中。在启动进入汉字系统后，选择执行下页二段程序之一，便能在MX-80Ⅱ型打印机上打印汉字。但前4种字型不能使用，即执行POKE1659，N命令时，n值必须在5-15之

间。

程序一：

```
5  REM MX-80 II CHANGES LFCODE
10  POKE 916,27：POKE 918,65：POKE 919,8
:POKE 920,8
```

程序二：

```
SOURCE FILE：CHANG1
0000:                    1 * MX-80 II CHANGES LF
CODE
0000:                    2 * * * * * * * * * * * * *
* * * * * * * *
-----NEXT OBJECT FILE NAME IS CHANG
OBJ0
1000:                    3              ORG       $1000
0394:                    4 LFCODE0    EQU       $394
0396:                    5 LFCODE2    EQU       $396
```

修改打印密度

位图象打印控制命令格式为ESCXn1n2，其X可以是“K”或“L”。分别表示标准密度打印和倍密度打印。在汉字打印驱动程序中，选用的是倍密度打印。若希望使用标准密度打印，则需修改TPCODE单元的内容。在进入汉字系统后。用户程序中只要执行语句POKE915，75便可实现打印密度的转换。

上述两种修改，退回到文本显示状态，再重新进入汉字系统时，只要没破坏1147（$47B）单元的内容，其修改是不会被破坏的。

习 题

1. 上机实习拼音输入法。

2. 上机实习国标输入法。

3. 上机实习特殊字符输入法。

4. 试编一个打印各种字型汉字的程序上机通过。

5. 中华学习机能连接哪几种打印机打印汉字？哪些打印机需要修改才能打印汉字？

6. 试修改打印密度后再运行4题中的程序，对比其输出的汉字。

第十四章 LOGO语言

本章介绍中华学习机上LOGO语言的基本概念，作图方法，数、字和表的处理以及数值计算功能。

14.1 LOGO语言概述

14.1.1 什么是LOGO语言？

LOGO语言是美国麻省理工学院人工智能实验室于一九六八年开发的适合于青少年和初学者学习和使用的一种电脑语言。LOGO语言有很强的作图功能，能够进行比较复杂的字和表的处理，同时也能够用于进行一些数值计算工作。在国外，LOGO语言已相当流行。一般的微型机，如IBM PC、APPLE及其兼容机等均配备有LOGO语言。

"LOGO"这个词，原意是"符号"、"文字"或"思考"、"想法"的意思。设计者用LOGO来作一种电脑语言的名称，真可谓用心良苦，含义深刻。LOGO语言的主要发明者之一的佩帕特（Seymour Papert）教授认为：刻板的计算机辅助教学（CAI）实际上是用程序设计去束缚孩子们的想象力和创造力。六十年代对儿童进行计算机辅助教学的实践，也使人们认识到：辅助教学的目的，不应该是仅仅让计算机来教育儿童，更重要的是要让计算机帮助儿童学会如何思考问题和解决问题。LOGO语言采用孩子们喜闻乐见的用积木拼图的方式，通过所谓的海龟作图来学习编制

程序，训练青少年的逻辑思维和创造能力。

佩帕特认为，计算机会同电视机一样普及，因而他们研制的LOGO语言不仅适用于青少年，也适宜于非计算机专业人员学习和使用。即使缺乏数理知识的人，凭着直观感觉也能够把握计算机，从而理解程序设计。

LOGO语言在国内也有了较快的发展。除了IBC PC、APPLE Ⅱ 等机器配备有LOGO语言外，目前流行的中华学习机CEC-Ⅰ上专门固化有LOGO语言的一个子集。这就为我们学习和使用LOGO提供了方便的硬软件保证，也为我国大力普及LOGO语言创造了良好的条件。为什么要向青少年和初学者推荐LOGO语言呢？因为LOGO语言有如下一些特点：

1. 语句简单，易学、易用、易记。

LOGO的语句（或称为命令）可能是所有语言中最简单而最易学最易用的了。很多命令就是一个英文单词（部份单词可以缩写），用法显然。

2. LOGO是面向过程的语言。

过程可以由一个或多个LOGO命令组成。根据需要，用户可以定义自己的过程，并能够象LOGO的基本命令一样在程序中使用。

3. 交互式使用。

不论是LOGO的基本命令还是用户定义的过程，都可以直接在键盘上输入并立即执行，执行的结果立即在屏幕上显示出来。

4. 具有数、字和表的处理功能。

LOGO不仅提供了简单的数和字处理功能，还能够对比较复杂的表进行处理。

5. 过程的递归调用。

LOGO语言最具特色之一是过程的递归调用，这对编制结构化的高水平程序十分有用。

6. LOGO能够训练孩子们的想象力和创造力。

14.1.2 如何启动LOGO语言系统？

中华学习机CEC-Ⅰ上可以使用两种版本的LOGO语言。一是固化在主机上的LOGO语言，我们称为固化LOGO；另外还可以使用APPLEⅡ上的LOGO版本，它保存在一张软盘上，我们称为APPLE LOGO。两种版本的用法基本相同，但仍然有少许差别。

1. 启动固化LOGO的方法

在启动LOGO之前，假设硬件系统已经连接好了。

①打开屏幕电源。

②打开主机电源，系统自动进入西文BASIC状态，屏幕显示“］”提示符，再敲入LG↵

```
ZHONG  HUA  XUE  XI  JI
      VERSION  1.1
]LG  ↵
```

屏幕显示如下：

当屏幕出现“？”提示符时，表示机器已经处于LOGO状态，这时就可以敲入LOGO命令了。

```
CHINESE  EDUCATION  COMPUER
            LOGO
        VERSION  1.1
        1987.6
?
```

2. 用APPLE LOGO软盘启动系统

①将LOGO软盘轻轻地插入到驱动器中，然后关上驱动器的“门”。（软盘有商标的那一面朝上，注意不要插反了）

②打开屏幕电源和主机电源。稍候，屏幕显示：

```
PRESS  THE  RETURN  KEY  TO  BEGIN
IF YOU HAVE YOUR OWN FILE DISKETTE,
INSERT  IT NOW, THEN PRESS RETURN
▬
(C) LOGO  COMPUTER  SYSTEMS  INC.1982
```

屏幕中间的“▬”表示光标，光标上面的那段英文的意思是：如果你有了自己的文件盘，那么现在就插入，然后敲回车键。敲回车键表示开始。光标下面是版权。系统在屏幕上显示这段话的目的是提醒用户在使用LOGO盘以前一定要复制备份盘，以防LOGO系统盘遭到意外的破坏。

③假设我们敲入回车键↵，屏幕又显示：

```
WELCOME  TO  LOGO  IA1.5

? ▬
```

提示符“？”表示机器已经处于LOGO状态了。以上称为冷启动。如果在插入软盘之前主机电源已经打开，我们也可以在“]”提示符下敲入：

PR# 6 ↵或IN# 6 ↵，这样仍然可以把软盘启动，使之处于LOGO状态。这称为热启动。

14.1.3 LOGO的主要概念

在详细介绍LOGO之前，先说明几个概念。

1. 什么是海龟？

在用LOGO语言来作图时，经常要提到所谓的海龟。什么是海龟呢？在“？”状态下，我们敲入DRAW命令，在屏幕的中心，我们可以看见一个白色的小三角形，其头朝上，这就是LOGO语言中所说的海龟。我们作图，就是通过海龟的爬行来实现的。也有人把它称为图龟。利用海龟的运动轨迹画出线段从而构成图形，这就称为海龟作图。海龟的初始位置如图14.1所示。

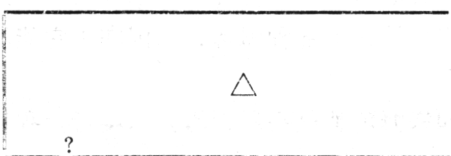


图14.1 海龟的初始位置

2. 海龟的活动范围

根据LOGO命令的指使，海龟能够在屏幕内上下左右移动，其活动范围用座标示于图14.2。

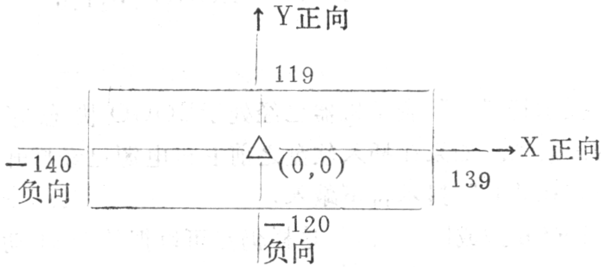


图14.2　海龟的活动范围

海龟的起始位置在屏幕的正中（0，0）。从起始位置，海龟向上最多爬行119步，向下最多爬行120步，向左最多爬行140步，向右最多爬行139步。一般来说，海龟就在这个范围内活动。在屏幕上绘图时最好限制在这个范围内。

如果海龟的活动超出了上述的界限，就自动构成循环爬行。就是说，如果海龟向上爬行超出了119步，那么就自动从对称的负方面（屏幕下端）钻出来。从而构成循环爬行。类似的，如果海龟在左端超界必然从右端出来，而右端超界必然从左端钻出来。从这点来说，海龟的活动范围又不受限进。

3．前进、后退、左转、右转

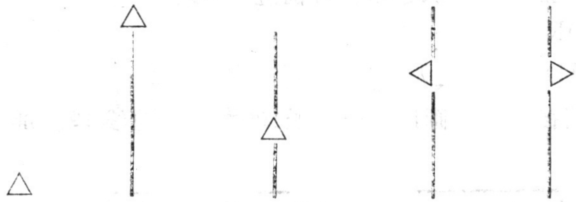
海龟能够方便地作出各种图形，原因在于它能够前后左右行动自如。

前进——如果海龟朝龟头所指的方向爬行，我们称为前进。如图14.3（a）。

后退——如果海龟朝龟头所指的相反方向爬行，我们称为后退，如图14.3（b）。

左转——如果海龟头朝反时针方向偏转，我们称为左转。如图14.3（c）。

右转——如果海龟头朝顺时针方向偏转，我们称为右转。如图14.3（d）。



起始位置　（a）　（b）　（c）　（d）

图14.3 海龟运动示意图

在上图中，海龟最先处于初始位置（屏幕中心），图14.3（a）表示海龟前进了50步，（b）表示海龟后退（在a的基础上）了30步，（c）表示海龟左转90度，（d）表示海龟右转90度。

在LOGO语言的叙述中，有时也用到｛…｝，<>、／、↲等符号，其含义与BASIC语言中约定相同。

14.2 LOGO的绘图功能

当系统启动屏幕显示“？”提示符后，我们就可以用LOGO命令正式工作了（最好先用固化LOGO比较简便）。

14.2.1 海龟作图的基本命令

1. 前进命令（FORWARD，缩写为FD）

格式：FORWARD／FD 〈参数〉

作用：指使海龟前进若干步，并画出一条线段（海龟爬行的足迹）。

这里的参数指出了海龟爬行的步数，或者说需要前进的距离。参数可以是正整数，也可以是负整数，当然还可以是带小数。

例14.1　指使海龟从起始位置前进100步。

? DRAW ↵

? FD　100 ↵

这就在屏幕上画出了一条长100海龟步的线段。屏幕显示如下：

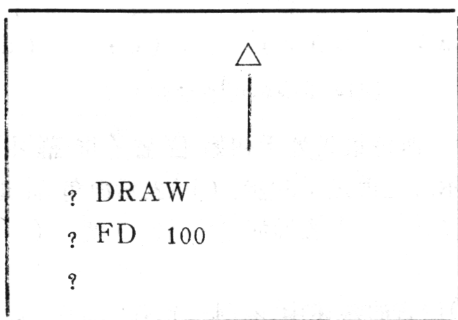


图14.4　海龟前进100步

DRAW命令清除屏幕并使海龟处于起始位置。“?”是提示符，不能敲入。

以上示例中，海龟是朝Y轴的正方向前进100步。如果海龟朝负方向前进100步，那又如何表示呢？请看下面的例子：

? DRAW ↵

? FD　-100 ↵

从该例看出，海龟实际是向下退了100步，龟头仍然指向上方。

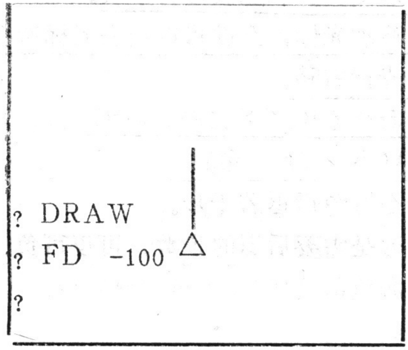


图14.5 海龟前进－100步

注意：

① FD和参数之间至少留一个空格。其它命令亦如此。

②一个行上可以输入多个命令，但不同命令之间必须用空格隔开。如：

? DRAW FD 100 ↵是正确的，而DRAWFD 100或DRAW FD100都是错误的。

③如果参数是负值，那么海龟就退步爬行。

④海龟可以在当前位置（不一定是初始位置）上继续前进。如

? FD 50 FD 30 FD -10 ↵这表示海龟先前进50步，再前进30步，又后退10步。

在输入LOGO命令时，如果字符拼写输入错了，那么屏幕提示信息：THERE IS NO PROCEDURE NAMED×××。这句话的意思是：不存在命名为×××的过程（过程的概念后面介绍）。只要屏幕上出现这种提示，就表示我们输入有误。改正错误的最简单方法是回到？后重新输入，因为LOGO命令较短，重新输入更快。

当然也可以利用◁、▷和△键进行修改。

◁键向左移动光标，△键可以删去光标所在行右端的字符。▷可以将光标右移。

2. 后退命令（BACK，缩写为BK）

格式：BACK／BK〈参数〉

作用：命令海龟后退若干步。

这里的参数是需要后退的步数，可正可负。

例14.2 先使海龟前进100步，再后退50步。

```
?   DRAW  ↵
?   FD  100  ↵
?   BK  50  ↵
```

屏幕显示如图14.6。

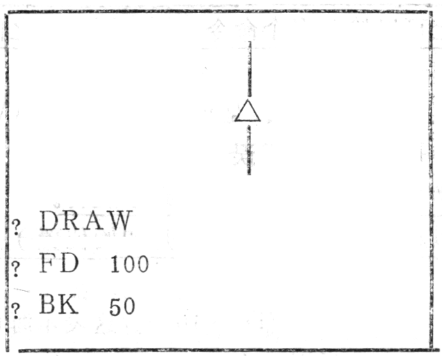


图14.6

注意：海龟后退时龟头并没有转向，仍然是龟头向上。如果参数为负值，情况又如何呢？

```
? DRAW  ↵
? BACK  -100  ↵
```

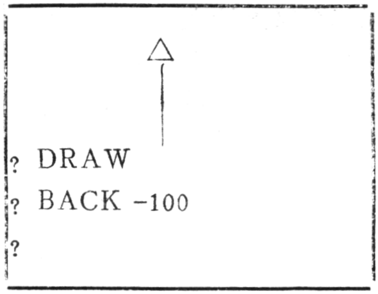


图14.7

从实际效果来看，BACK -100相当于FD 100，而BACK 50相当于FD -50。因此，前进和后退不是截然分开的且前进或后退的距离是用龟步来表示的。

3.，右转命令（RIGHT，缩写为RT）

格式：RIGHT／RT 〈参数〉

作用：指使龟头向右偏转若干度。

这里的参数是指转动的角度。可以是正数，也可以是负数。

例14.3 海龟前进50步，右转90度，再前进50步。所用命令如下：

```
? DRAW   ↵
? FD 50  ↵
? RT 90  ↵
? FD 50  ↵
```

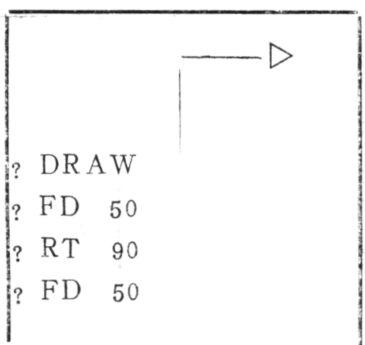


图14.8

执行命令的结果如图14.8所示。

如果转动角度是负数，那情况如何呢？

```
? DRAW ↵
? FD  50 ↵
? RT  -90 ↵
? FD  50 ↵
```

执行结果如图14.9所示。

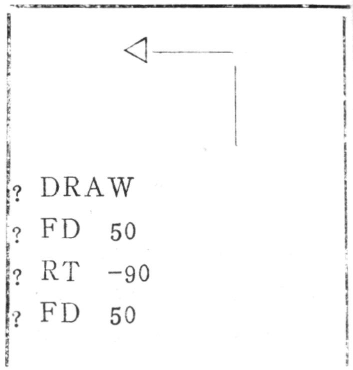


图14.9

4. 左转命令（LEFT，缩写为LT）

格式：LEFT／LT 《参数》

作用：指使海龟头向左偏转若干度。

例14.4 海龟前进50步，左转90度后再前进50步。

```
? DRAW ↵
? FD  50 ↵
? LT 90↵
? FD  50 ↵
```

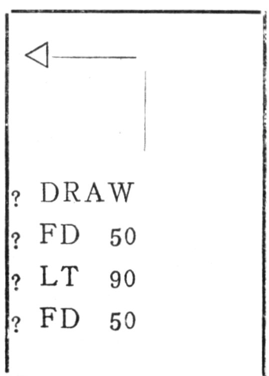


图14.10

执行结果如图14.10所示。

左转角度是负数的情况。

? DRAW ↵

? FD 50 ↵

? LT -90 ↵

? FD 50 ↵

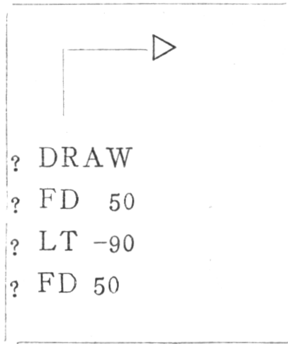


图14.11

执行结果如图14.11所示。

从上面的图示可以看出：RT 90等于LT -90，RT -90等于LT 90。对于FD和BK命令，它们的参数是用海龟步来表示的，而RT和LT命令，其参数均表示转动的角度。如果给出的角度大于360，那么转动的实际度角就是去掉若干个360（一个圆）以后剩下的度数。如RT 380，实际相当于RT 20。

5. 清屏幕命令

格式：DRAW／CLEARSCREEN／CS／HOME

作用：这几个命令与清屏幕有关。

DRAW——清屏幕后把海龟确定在起始位置。固化LOGO中用DRAW是很方便的，APPLE LOGO中有时不能见效。

CLEARSCREEN或CS——只清除屏幕上的字符和图形，不去掉海龟。CS是CLEARSCREEN的缩写，其作用相同。两种版本均可使用。

HOME——将海龟从其它位置挪回到初始位置，并画出线段。

例14.5　在屏幕上图一个正方形。

```
? DRAW ↵
? FD 80 ↵
? RT 90 ↵
? FD 80 ↵
? RT 90 ↵
? FD 80 ↵
? RT 90 ↵
? FD 80 ↵
```

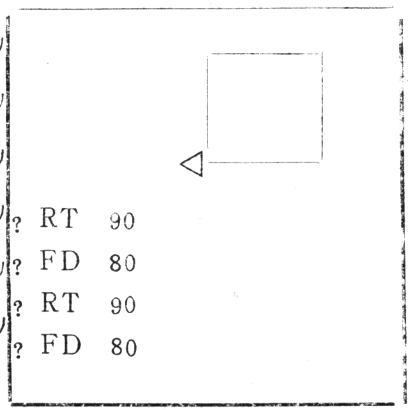


图14.12

执行结果如图14.12所示。为什么屏幕上只显示了最后四条命令呢？原因是我们现在所处的方式正是LOGO的绘图模式方式。这种方式使得屏幕上部用于绘图，下部可以显示至多四条命令。

请读者将参数取为负值以及再换后退、左转等命令画出同样大小的正方形。注意这些正方形在屏幕上的位置有什么不同。

现在，我们能否只用FD和RT两种语句（使用次数不限），画出正三角形，正五边形，甚至一个近似的圆？请读者先在机器上试试看。如果要画长方形和任意边长的三角形呢？

6． 隐藏或显示海龟

格式：HIDETURTLE／HT

作用：将海龟隐藏起来，使之不显示在屏幕上。

这个命令不带有任何参数。有时为了使画出的图形美观，不致因海龟的出现而令人扫兴，我们就可以将海龟隐藏起来。

例14.6 画一个等边三角形。

```
? DRAW ↵
? FD  50 ↵
? RT  120 ↵
? FD  50 ↵
? RT  120 ↵
? FD  50 ↵
? HT ↵
```

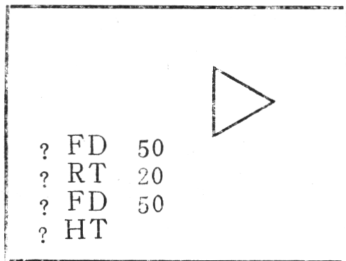


图14.13 等边三角形

执行结果如图14.13。

HT的作用就是将海龟藏起来，因此从图上就看不见海龟的踪影了。

格式：SHOWTURTLE／ST

作用：将隐藏的海龟显示出来。

如果我们需要把上图中的海龟显示出来，那么只需要敲入ST↵就行了。

7． 重复命令(REPEAT)

在前面画正方形和三角形的例子中我们已经看到，相同的命令重复了多次。如果程序中某些命令需要成千上万次的重复，那是不可能逐条写出所有命令的。这就必须求助LOGO语言中提供的重复命令。

格式：REPEAT 〈参数〉［〈命令序列〉］

作用：将方括号中的命令重复执行若干次。

其中：参数是需要重复执行的次数，它只能是一个正数。命令序列可以是一个或多个命令，如果有多个命令，那么各命令彼此间必须用空格隔开。

例14.7　用REPEAT命令画一个正方形。

```
? DRAW ↵
? REPEAT  4 [FD  80  RT  90] ↵
```

执行这个命令就画出了图14.14所示的正方形。在上述命令中，方括号中的“FD 80 RT 90”重复执行了四次。请仔细观察这个正方形绘完后龟头的指向同前面的正方形有何不同？



图14.14

例14.8　画一个等边三角形。

```
? DRAW ↵
? REPEAT 3[FD  50  RT  120] ↵
```

例14.9　画一个正五边形。

```
? DRAW ↵
? REPEAT 5 [FD 30 RT 72] ↵
```

请读者按照以上两例的描述，画出图形，特别注意海龟的位置和方向。

例14.10　画出如图14.15所示的一个风车。

在没有给出程序之前，我们先来观察这个风车有些什么特点，然后再确定用哪些命令来实现。由图中可以看出风车有以下规律：

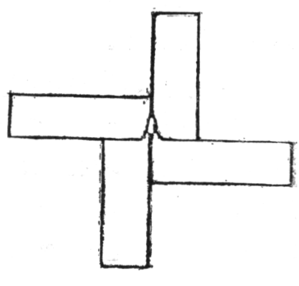


图14.15 风车

①由四个长方形组成。

②如果以Y轴正向的长方形为准，那么后一个长方形的位置只是在前一个长方形的基础上向右转动了90度。

我们知道画长方形的命令可以是：

FD 60 RT 90 FD 20 RT 90重复两次，即

REPEAT 2〔FD 60 RT 90 FD 20 RT 90〕，再将整个长方形的位置转动90度，就又生成了第二个长方形。如此重复四次，风车图形就画出来了。将上述说法用LOGO命令写出来就是：

```
? DRAW ↲
? REPEAT 4〔REPEAT 2〔FD 60 RT 90
FD 20 RT 90〕 RT 90〕 ↲
```

上述程序完全是向右转动角度，请读者用左转或负值画出同样的风车图形。

例14.11 画出图14.16所示的一个魔球。

初看起来，这个图形是相当复杂的。但我们仔细观察后就会发现，魔球是由若干个正五边形旋转一定角度组成的。有多少个正五边形呢？实际数一数就知道是10个。也就是说，最外面的那个正10边形是由一个正五边形每次旋转360/10度，共旋转10次而组成的。把这两点看清楚了就不难给出绘图命令。

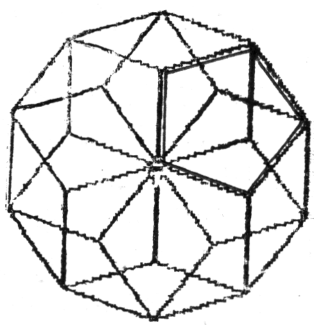


图14.16 魔球

```
? DRAW ↲
? REPEAT 10〔REPEAT 5〔FD 40 RT 72〕
RT 36〕 ↲
```

最内层的重复命令执行5次后就画出一个正五边形，然后再将下一个正五边形的起始位置右转36度，如此重复10次，一个魔球就画出来了。

从以上例子我们可以得出结论：如果要画一个封闭的正多边形，那么海龟转动的角度之和应该是360的整倍数。否则就构不成一个封闭的多边形。

8．抬笔／落笔命令

至此，我们已经能够画一些比较复杂的图形了。但是，如果需要画一条虚线，那就把我们难住了。下面介绍的抬笔落笔命令能够为我们解难。

格式：PENUP／PU

作用：命令海龟抬起画笔，使得海龟的运动轨迹不显示在屏幕上。

格式：PENDOWN／PD

作用：命令海龟落下画笔，使得海龟的踪迹显示在屏幕上。

通过海龟画笔的一起一落，就能画出虚线来。

例14.12 画一个由虚线组成的正方形。

```
? DRAW ↲
? REPEAT 4 [REPEAT 10 [FD 3 PU
FD 2 PD] RT 90] ↲
```

最内层的重复命令执行10次后才能画出正方形的一条虚线边，只有最外层的4次重复执行完成后，整个正方形才全部画完。如图14.7。



图14.17

注意：PU和PD命令通常联合使用，为什么？请读者考虑。

9．测定角度

格式：HEADING

作用：测定海龟从起始位置到现在位置（或称为当前位置）所转动的角度。

当海龟处于屏幕正中的起始位置时，其角度总是为0度。

例14.13 看下面的例子。

```
? DRAW ↵
? REPEAT  30 [FD  1  RT  1] ↵
? HEADING ↵
RESULT：30     （表示结果是30度）
```

如果我们朝左转呢？

```
? DRAW ↵
? REPEAT  30 [FD  1  LT  1] ↵
? HEADING ↵
RESULT：330     （结果是330度）
```

从上例可以看出：HEADING总是按顺时针方向且以起始位置为准来测定海龟转动的角度。

10．测定座标值

格式：XCOR／YCOR

作用：XCOR用来测定海龟当前位置的X座标值。YCOR用来测定海龟当前位置的Y座标值。

例14.14 参看下面的命令。

```
? DRAW ↵
? REPEAT  30 [FD  1  RT  1] ↵
```

? XOCR ↵

RESULT：7.42598（表示现在X座标值为7.42598）

? YCOR ↵

RESULT：28.7141（表示现在Y座标值是28.7141）

11. 测定Y轴正方向与已知点的角度

格式：TOWARDS 〈参数-1〉 〈参数-2〉

作用：测定Y轴与某个已知点座标位置之间的夹角是多少度。

其中：参数-1、参数-2分别是已知点的X座标和Y座标值。

例14.15 测定海龟的起始位置与（30，30）点的角度。

? DRAW ↵

? TOWARDS 30 30 ↵

RESULT：45.0006 （精确结果应该是45度，误差是由于屏幕纵横比例差造成的）

注意：①当命令中需要用到两个参数时，若第二个参数为负值，则必须用圆括号（ ）将参数括起来。如TOWARDS 30 （-30）。②该命令与海龟头的指向无关。

12. 海龟的水平移动

除了用FD和BK命令使海龟上下左右移动以外，下面的SET命令也能起到同样的作用。

格式：SETX 〈参数〉

作用：使海龟水平移动到指定的X座标值。

这里的参数就是屏幕上X座标的值。

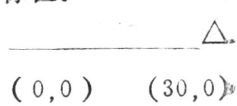


图14.18

例14.16 使海龟平移30步。

? DRAW ↵

? SETX 30 ↵

注意：参数为负值则向左移动，正值向右移动。

13．海龟的垂直移动

格式：SETY 〈参数〉

作用：使海龟垂直移动到指定的Y座标值。

这里的参数是屏幕上Y的座标值。

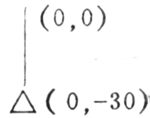


图14.19

例14.17 使海龟垂直移动30步。

? DRAW ↵

? SETY -30 ↵

注意：参数为正则向上移动，为负则向下移动。

14．移动海龟到给定的座标位置

格式：SETXY 〈参数-1〉 〈参数-2〉

作用：将海龟移动到指定的座标位置。

这里的参数-1和参数-2分别是海龟的X座标和Y座标值。

例14.18 将海龟从起始位置移到(30，50)位置。

? DRAW ↵

? SETXY 30 50 ↵

执行结果如图14.20。

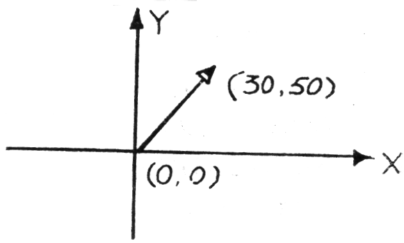


图14.20

我们再看下面几个座标位置。

? DRAW ↵

? SETXY 30 (-50) ↵

执行结果如图14.21.

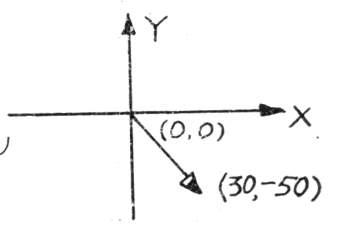


图14.21

? DRAW ↵

? SETXY -30 50 ↵

结果如图14.22。

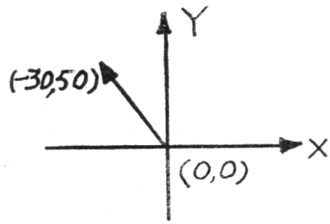


图14.22

? DRAW ↵

? SETXY -30 (-50) ↵

结果如图14.23。

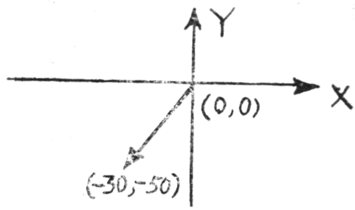


图14.23

注意：当第二个参数为负数时一定要用圆括号( )括起来。有了SET命令，移动海龟更为方便了。

15. 将海龟转动到指定的角度

格式：SETHEADING 〈参数〉

作用：将海龟偏转到指定的角度位置。

例14.19 将海龟从起始位置转动45度。

? DRAW ↵

? SETHEADING 45 ↵

FD 50 ↵

结果如图14.24。

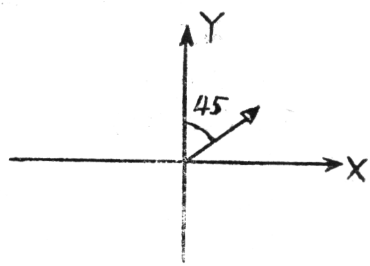


图14.24

? DRAW ↵

? SETHEADING -45 ↵

? FD 50 ↵

结果如图14.25。

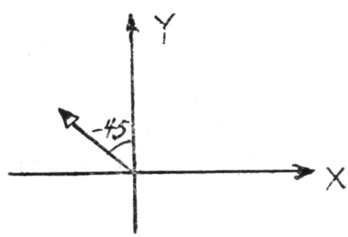


图14.25

注意：该命令以起始点为准，与海龟当前角度无关。

16. 设置纵横比例值

通过上机实习，读者可能已经发现了：我们在屏幕上绘出的正方形不太正，画出的圆也不太圆。这是什么原因呢？主要是屏幕的水平方向和垂直方向距离不等造成的。也就是说，海龟从水平方向前进一步比垂直方向前进一步距离要长一点（按理说应该相同）。通常屏幕上水平方向共有280个点，而垂直方向上只有240个点。这就造成画方不方画圆不圆。为减少这种失真度，LOGO提供了设值纵横比例值的命令。

格式：.ASPECT 〈参数〉

作用：设置屏幕上纵横比例值。

这里的参数通常设为0.9。如果我们启动系统后就立即设置比例值，那么直到重新启动系统之前，就一直用该比例值进行工作。

例14.20 将纵横比例值设为0.9。

```
? .ASPECT 0.9↵
```

注意：该命令之前必须带一个小数点。

请读者用画圆来试验，当给出不同的比例值时，画出的圆有多大差别。

17. 如何打印屏幕图形？

①正确连接好打印机线路，打开其电源。

②如果打印机的接口卡插在1号槽口，就敲入：

```
? OUTDEV 1 ↵
```

③设置打印参数，其一般格式是：

```
DEPOSIT 1913 X
```

其中：X＝

1	正常打印
33	反相打印
65	放大打印
97	反相放大打印

如果省略该命令，则表示按正常打印。

④屏幕硬拷贝打印，敲入：

？CHAR 17↙

注意：这几个命令在固化LOGO中十分有效，但在APPLE LOGO中CHAR 17却不起作用，原因不太清楚。本章中的所有图形均是在固化LOGO版本下用CHAR 17打印出来的。打印机型号是SP－1000。

14.2.2 过 程

前面介绍了LOGO的基本作图命令。如果只能直接敲入基本命令，那么LOGO语言的应用将受到很大的限制。下面介绍如何用过程来扩充基本作图命令。

1. 什么叫过程？

简单地说，把若干命令汇集到一起所组成的一个程序段，我们都可以看成是一个过程。对于一个过程，必须向系统提供以下信息：

①定义过程的标志是什么。系统如何才能判断我们需要建立一个过程？LOGO语言用TO作为建立过程的标志。

②过程的名字是什么。即是说必须给过程取一个名字。LOGO规定，过程名可以由字母，数字等组成，但过程名中不能含有空格、括号或算术运算符（＋－ *∕）。

③有些什么参数。即过程是否需要同外界交换信息。如果需要改变过程中某些命令的运行情况，那么就必须给出过程参数。否则，可以没有参数。

④过程由哪些命令组成，即必须列出过程中用到的全部命令。

⑤过程定义的结束标志。

2. 过程的建立方式

格式：TO 〈过程名〉 〔参数—1，参数—2…〕

〈过程体〉

END

作用：告诉计算机系统准备建立一个过程。

其中：TO——建立过程的标志。过程名是用户给自己定义的过程取的名字。根据需要，可以带有若干个参数。过程体是完成某些操作的若干命令序列。END是过程的结束标志。END不是必需的，可以去掉。但以后示例中我们均带上，以表明过程完结。(APPLE LOGO用END作为过程结束标志)

由于参数是根据需要来选定的，若去掉参数任选项，那么这种过程就称为无参过程，又称简单过程。带有参数的过程称为有参过程。参数的个数根据实际需要而定。

一个已经定义的过程，LOGO可以把它当成一个基本命令来使用，如同增加了一个新的命令一样。

我们用例子来说明如何定义过程。

例14.21 将画正方形的命令用过程来定义。

```
? TO SQUARE↲
  DRAW↲
  REPEAT 4 〔FD 60 RT 90〕↲
  END
```

(在APPLE LOGO中，END就表示定义过程结束，因而不必敲CTRL-C键)

CTRL-C（在固化LOGO中，必须敲CTRL-C键才会结束过程定义）

这就定义了一个名字为SQUARE的过程。这是一个无参过程。过程体包含了两个命令。过程的定义情况是这样的：

当我们在“？”之后敲入TO SQUARE↲时，整个屏

幕清为空白，然后上方出现？TO SQUARE，屏幕的下方出现提示：

EDIT：CTRL-C TO DEFINE，CTRL-G TO ABORT

这个提示的意思是：定义过程时我们可以使用编辑功能键CTRL-C和CTRL-G。

CTRL-C的作用是结束过程的定义，使过程有效。

CTRL-G的作用是结束过程的定义，使过程无效。

我们依次敲入需要的命令。这种情况下，命令前面的问号就不再出现了（？可以看成是对话使用LOGO命令的标志）。一个有效的过程定义结束后，一定要记住敲CTRL-C键（在固化LOGO版本中必须如此）。为了告诉用户过程定义是否成功，系统在屏幕上显示：

PLEASE WAIT…	（请稍候…）
SQUARE DEFINED	（SQUARE过程已定义）
？	（又回到对话方式）

至此，一个过程才算定义完毕。

如果我们需要调用SQUARE过程画出正方形，只需在？之后敲入过程名就是了。如：

？SQUARE ↵

执行结果如图14.26所示。

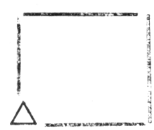


图14.26　正方形

3. 过程的编辑

在定义过程时，如果发生了拼写错误，有两种方法来更正。一是当还没有退出过程定义时就发现某个字符输入错了，那么可以用◁键回退到该字符，重新输入正确字符。二

是已经退出过程定义后才发现输入有误，那就只好用过程的编辑命令了。

我们先介绍编辑（修改）过程时可用的一些光标控制键（在固化LOGO中很有效，但APPLE LOGO中大部份不起作用）：

键	功能
ESC	删去光标左边的那个字符，右边字符依次左移一个字符位置
△	删去光标所在行右端的字符。如果光标右端是空行，则将下行字符上提。
▷	光标右移一个位置。
◁	光标左移一个位置
CTRL-A	光标移到一行的开头。
CTRL-B	光标上移一页，若不足一页则移到最开头。
CTRL-C	定义过程，退出编辑状态，回到对话方式，修改有效。
CTRL-D	删除光标所在字符，右端字符依次左移一格。
CTRL-E	光标移到行尾。
CTRL-F	光标下移一页，不足一页时移到最末尾。
CTRL-G	退出编辑状态，回到对话方式，修改无效。
CTRL-H	相当于◁键。（左箭头键）
CTRL-K	删去光标所在行右端的字符。相当于△键。
CTRL-M	相当于↵键，将光标所在行下推，从而空出一行。

CTRL-N　　光标下移一行。

CTRL-O　　从光标位置的下一行开始插入空行。

CTRL-P　　光标上移一行。

我们还要介绍一个编辑命令：

格式：EDIT/ED 〈过程名〉

作用：在屏幕上显示出过程名及其全部文本字符，允许用户对文本字符进行编辑和修改。

①插入一个新行

将光标移到待插入新行的位置，敲↵键（或者CTRL-M键），这时屏幕上就空出一行，再敲CTRL-P键，把光标移到空行上，敲入新插入的命令就行了。

②删除一行

将光标移到待删除行的开头，敲△（向上箭头键）键就把该行的所有字符去掉了，从而留下一个空白行。

③在已知字符中插入一个新字符

将光标移到（可用CTRL-P上移一行，或用CTRL-N下移一行）待插入字符的位置上，敲入新字符，原来的字符就自动右移。

在编辑过程时，如果希望编辑生效，那么修改结束后一定要敲CTRL-C键。如果希望修改作废，那么就敲CTRL-G键，过程仍然保持原来的面目。

4．如何将LOGO程序存盘？

①存盘命令

格式：SAVE　“文件名

作用：将内存的一个或若干个过程保存到软盘中。

其中：文件名是用户给需要保存的若干过程取的名字。

例：假设我们在内存中建立了S1，S2，S3，S4，S5等

5个过程。

? SAVE "HAN1↲

这就把S_1，S_2，S_3，S_4，S_5用HAN1文件名统起来保存在软盘上了。

②读盘命令

格式：LOAD "文件名

作用：将磁盘上指定的文件读入内存。

例：? LOAD "HAN1↲

这就把过程S_1，S_2，S_3，S_4，S_5又读入到内存，根据需要可以马上执行这些过程。有了这两条命令，保存程序就十分方便了。

③列磁盘目录

格式：CATALOG

作用：列出磁盘上的所有文件名，带有LOGO后缀的是用户文件。

④显示过程内容

格式：PO "过程名

作用：在屏幕上显示出指定过程的内容。

例：? PO "SQUARE↲

⑤显示所有过程

格式：POTS

作用：显示内存中的所有过程名。

如果显示的过程名较多，敲CTRL－W键使显示暂停，敲任意键又继续后继显示。

⑥删除过程

格式：ERASE/ER "过程名

作用：将指定的过程名删去。

注意：以上命令对APPLE LOGO很有效，而对固化LOGO．SAVE和LOAD均不起作用。

5． 无参过程

在建立过程时如果没有带任何参数，那么这种过程称为无参过程。无参过程只能够严格地按过程体内规定的距离和角度进行操作，它既不能缩小，也不能放大，而且每次执行过程都是如此。

例14.22 画出图14.27所示的图形。

仔细观察图形后就会发现，该图是由一个正方形旋转45度后重复8次构成。

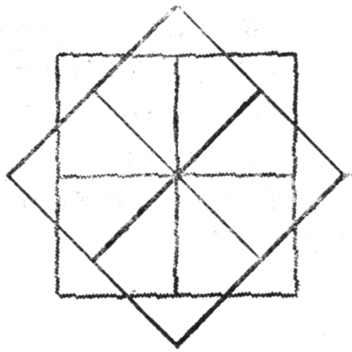


图14.27 E14.221

用过程写出程序：

```
? TO E14.221↲
DRAW↲
REPEAT 8〔REPEAT 4 〔FD 60 RT 90〕 RT
45〕↲
HT↲
END↲
CTRL-C

?
```

过程定义完毕。我们现在来调用过程以画出图形。

```
? E14.221↲        （调用E14.221过程）
```

执行结果如图14.27所示。

我们也可以把画正方形的命令用过程的形式写出，然后在E14.221过程中调用之。如：

```
? TO SQU↙
REPEAT 4 [FD 60 RT 90] ↙
END↙
CTRL-C
? TO E14.222↙
DRAW ↙
REPEAT 8 [SQU RT 45] ↙
HT ↙
END↙
CTRL-C
? E14.222↙
```

调用后画出图形同图14.27一样。

在E14.222过程中，我们调用了SQU过程，这是允许的。前面已经提及，一个过程一经定义，就可以象LOGO的基本命令一样地使用，这给LOGO增色不少。

6. 有参过程

如果我们希望画出不同大小的圆，那么可以利用有参过程来实现。

例14.23 画出大小不同的三个圆。

我们用正36边形来近似地代表一个圆，要使圆的大小不同，无非是要变化正36边形的边长，我们用下面的有参过程来实现：

```
? TO E14.23 :R↙
REPEAT 36 [FD :R
RT 10] ↙
END↙
CTRL-C
? E14.23 3 ↙
```

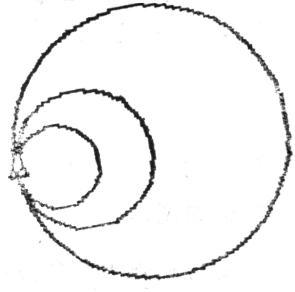


图14.28 不同大小的圆

? E14.23　　5↙

? E14.23　 10↙

执行结果如图14.28。

在上述程序中，:R是有参过程E14.23的参数，它表示在调用过程时该参数的大小是可以变化的。其实际含义就是正36边形的边长可以根据需要而取3或5或10，因而画出的圆大小也就不同了。在调用时，参数所在的地方要给出一个具体的值。如E14.23 3，表示用3去代替过程体中出现的:R，然后执行之。E14.23 5 和E14.23 10类似。

上例中只有一个参数。事实上，根据需要可以设立多个参数。

例14.24 画出不同边长的长方形。

我们将长方形的长和宽分别用参数:L和:W来表示。

```
? TO E14.24   :L  :W↙
REPEAT 2 [FD :L RT 90 FD :W RT 90] ↙
END↙
CTRL-C
```

下面调用E14.24有参过程画出图形。

? DRAW↙

? E14.24 50 20↙

结果如图14.29。

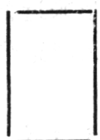


图14.29 E14.24 50 20

? DRAW↙

? E14.24 20 50↙



结果如图14.30。

图14.30 E14.24 20 50

在编写和调用有参数过程时必须注意：

①参数之前必须紧跟一个冒号。如 :L， :W。并且冒号和参数名之间不能留空格。

②调用有参过程时对应的实在参数之前不能带冒号。如：E14.24 50 20是正确的，而E14.24 :50 :20是错误的，原因是50，20之前有冒号。

我们再举两个有参过程的例子。

例14.25 如何画圆弧。

画圆弧要考虑三个因素：一是每次移动的距离，二是转动的角度，三是重复执行的次数。这里将每次转动的角度固定，把移动距离和重复次数作为过程的两个参数。

```
? TO E14.251 :D↵
REPEAT :D [FD 3.14159/180* :R RT 1] ↵
END↵
CTRL-C
```

调用E14.251过程：

```
? DRAW ↵
? E14.251 60 80↵
? DRAW↵
? E14.251 100 90↵
```

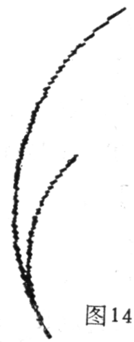


图14.31 弧

执行结果如图14.31所示。上述过程虽然能够正确画出一段圆弧，但速度太慢。原因在于每次转动的角度太小。我们将上述程序改进一下，将每次转动的角度扩大10倍。

```
? TO E14.252 :R :D ↵
REPEAT :D/10 [FD 3.14159/18* :R RT 10] ↵
END ↵
CTRL—C
```

调用E14.252 过程。

? E14.252 60 80 ↲

? E14.252 100 90 ↲

执行结果同图14.31一样。这里的参数R和D，实际就是圆弧的半径和转动角度。需要注意的是：如果重复次数带有小数，那么重复次数只取整数值。

例14.26 如何画树叶。

我们可以把树叶看成是由右弧和左弧封闭而成。先编写画左、右弧的程序：

```
? TO RARC :R :D ↲      (画右弧)
REPEAT :D/10 [RT 5 FD 3.14159/18* :R RT
5] ↲
END ↲
CTRL-C
? TO LARC :R :D ↲ (画左弧)
REPEAT :D/10 [LT 5 FD 3.14159/18* :R
LT 5] ↲
END ↲
CTRL-C
? TO E14.26R :R :D
(向右斜树叶)
HT ↲
RARC :R :D ↲
RT 90 ↲
RARC :R :D ↲
END ↲
CTRL-C
```

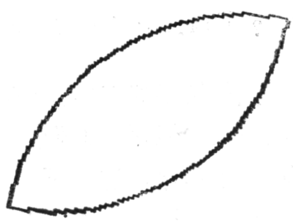


图14.32 右斜树叶

```
? E14.26R 100 90 ↵
```
执行结果如图14.32所示。

```
? TO E14.26L  :R  :D ↵      （向左斜树叶）
HT ↵
LARC  :R  :D ↵
LT  90 ↵
LARC  :R  :D ↵
END ↵
? E14.26L  100  90 ↵
```
图形如图14.33所示。

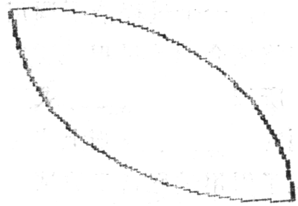


图14.33　左斜树叶

```
? TO E14.26RL :L :D ↵  （左右两片树叶并蒂）
E14.26R  :R  :D ↵
HOME ↵
E14.26L  :R  :D ↵
END ↵
? E14.26RL  100  90 ↵
```
（":L" 上方有手写的 "R"）

执行结果如图14.34所示。

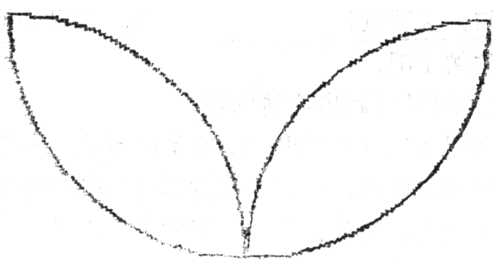


图14.34　并蒂树叶

14.2.3 递归过程和条件语句

1. 什么叫递归？

从前面的示例中我们已经看到了，一个过程可以调用另一个过程，这种调用可以一层一层的嵌套下去。那么读者可能会问：一个过程能否调用自身呢？当然是可以的。这就是我们将要介绍的过程的递归。

所谓递归，就是把求解问题的过程归结为对一系列已知问题的求解。从直观上来说，如果在一个过程的定义内部又出现了引用过程自身的情况，我们就把这种情况称为过程的递归。

举一个数学上应用递归的例子。

例如：求n！＝n(n－1)(n－2)…1的值，我们要求n！的值，用递归来处理就必须首先求出(n－1)！的值，而要求(n－1)！的值，又必须求出(n－2)！的值，如此递推下去，直到0！为止。假定0！的值已知为1。简单说来，用递归的方法来处理这类问题就是：求当前值必须先求出其前一项的值，这样逐项递推，直到寻“根”为止。再用递推反过来求当前值。这是数学上求递归运算的办法。

在LOGO语言中，如果一个过程中包含了过程自身，那么就称为递归，这种过程就称为递归过程。我们还是来看一个递归过程的例子吧。

例14.27 用递归过程画出旋转图案。

仔细观察图案后可以看出：图案是由不同大小的正方形旋转一定角度后组成的。因此，正方形的边长和角度要随着程序的执行而变化。另外，后一个正方形的生成完全依赖于前一个正方形，这就要用到前面介绍的递归的概念。过程中用L来表示正方形的边长。

```
? TO E14.27  :L ↵
   （画旋转图案）
SQUARE  :L ↵
RT  20 ↵
E14.27  :L+5
END
?TO SQUARE :L↵
   （画正方形）
REPEAT 4 [FD :L
   RT  90]
END
```

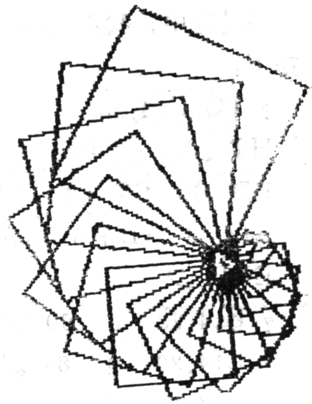


图14.35 旋转图案

? E14.27 5 ↵   结果如图14.35所示。

说明：在E14.27过程中，首先调用SQUARE过程画出第一个正方形；其次，将下一个正方形的方向右转20度，这样使得前后两个正方形不会重迭在一起。将正方形的边长增加5以后递归调用E14.27过程。如此反复执行就画出了我们所需要的图案。

如果读者亲自上机一试就会发现：程序的执行老是不能自动停下来。这是何故呢？这是由于递归调用中没有给出结束条件，因而使程序的执行没有尽头。我们可以敲CTRL—G键使之中止。记住：凡是需要程序中途停止运行均可用CTRL-G键。从上例可以清楚地看到：递归过程的特征就是过程体内又出现了过程本身。

2. 递归过程实例

递归过程用得相当普遍，而且给出不同参数所绘制的图形相差甚大。下面给出一个绘制旋涡线图案的递归过程。

例14.28　写出绘制旋涡线图案的递归过程。

在过程中，我们用L表示长度，用A表示转动角度。过程如下：

```
? TO E14.28  :L  :A ↵
HT ↵
FD :L  RT :A ↵
E14.28  :L+3  :A ↵
END ↵
? E14.28  0  90 ↵      图14.36。
? E14.28  0  89 ↵      图14.37。
? E14.28  0  91 ↵      图14.38。
? E14.28  0  118 ↵     图14.39。
? E14.28  0  119 ↵     图14.40。
? E14.28  0  72 ↵      图14.41。
? E14.28  0  73 ↵      图14.42。
? E14.28  0  143 ↵     图14.43。
? E14.28  0  144 ↵     图14.44。
? E14.28  0  190 ↵     图14.45。
```

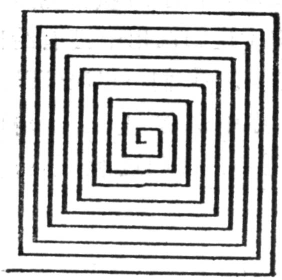


图14.36 E14.28　0　90

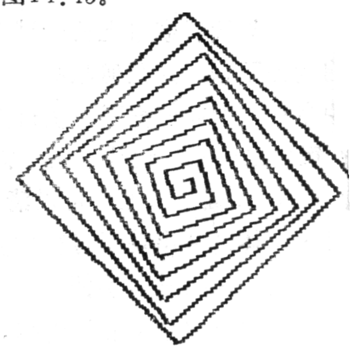


图14.37 E14.28　0　89

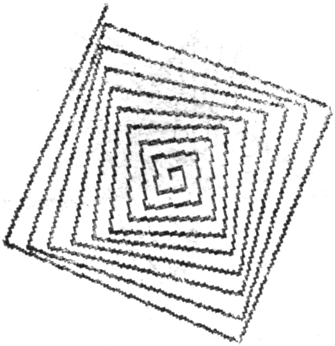


图14.38 E14.28 0 91

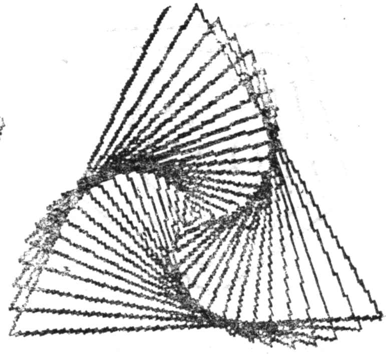


图14.39 E14.28 0 118

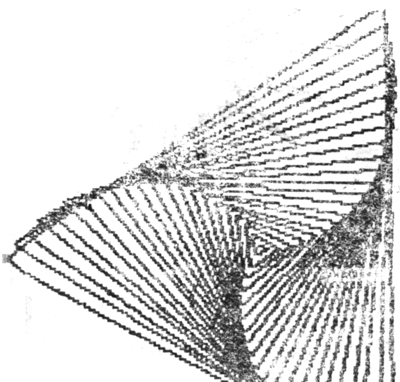


图14.40 E14.28 0 119

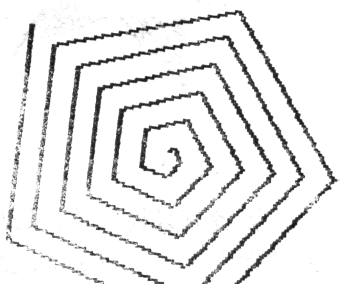
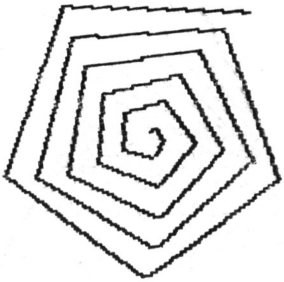
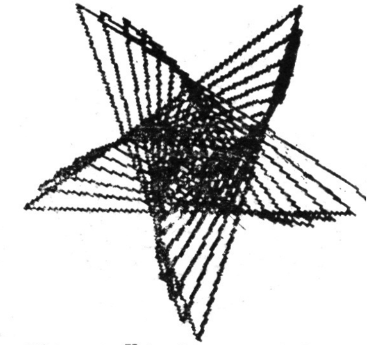
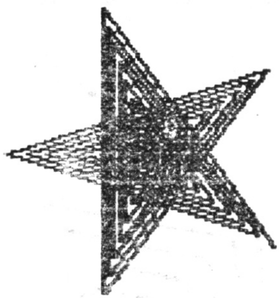
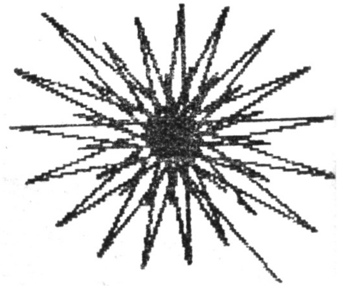


图14.41 E14.28 0 72


图14.42 E14.28 0 73


图14.43 E14.28 0 143


图14.44 E14.28 0 144


图14.45 E14.28 0 190

读者有兴趣的话，还可以给出一些参数值进行调用，得到的图形是非常有趣的。从以上可以看到：程序十分简单，而作出的图形却相当复杂，由此可见递归过程的魅力了。

例14.29　在例14.28中，随着递归过程的执行，海龟移动的长度也在随之变化，而角度却没有变化。如果我们反过来变化角度而不变长度，其绘出的图案又如何呢？请看下列程序：

```
? TO E14.29  :L  :A ↲
HT ↲
FD  :L  RT  :A ↲
E14.29  :L  :A+10 ↲
END
```

我们现在执行递归过程：

```
? E14.29  12  3 ↲      图14.46。
? E14.29  10  5 ↲      图14.47。
? E14.29  15  1 ↲      图14.48。
? E14.29  15  2 ↲      图14.49。
```

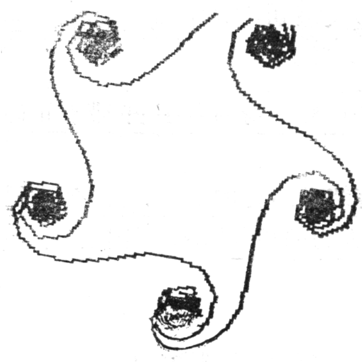


图14.46 E14.29 12 3

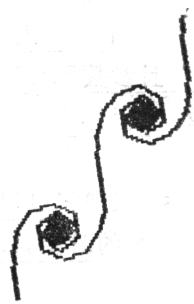


图14.47 E14.29 10 5

在图14.46中，由于右上方图形超出了屏幕，因而没有显示出来。



图14.48　E14.29　15　1

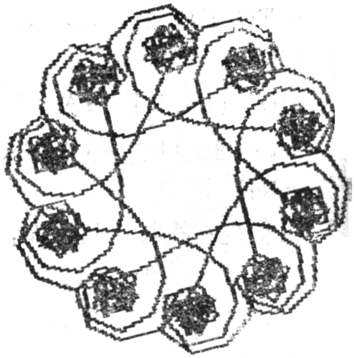


图14.49　E14.29　15　2

读者还可以给出一些参数来运行该过程。对于递归过程，我们要注意以下几点：

① 递归过程中允许有多个参数。参数之前应带有冒号“:”且参数和:之间不能有空格。

② 过程体中一定要出现过程本身的名字并带有正确的参数。

③ 如果递归过程中没有结束运行的命令，那么当程序执行到适当的时候我们可以人为的用CTRL-G键来停止程序的执行。

3． 条件语句和停语句

为使程序执行到满足某种条件后自动停止，LOGO语言提供了条件语句和停语句。根据给出条件的形式不同，条件语句又分为三种格式：

格式一1：IF 〈条件〉 [THEN] 〈命令序列〉

作用：当条件成立时就执行指定的命令序列。否则顺序执行后继命令。

其中：条件是由关系运算符＞、＝、＜构成的关系表达

式，它类似BASIC中的关系表达式。命令序列可以是一个或多个LOGO命令，也可以是过程名。THEN是待选项，可以写出来，也可以不写。

例如：IF :L<80 THEN SQUARE :L。

格式一2：IFTURE/IFT 〈命令序列〉

作用：如果测试条件为真，则执行命令序列。否则，顺序执行后继命令。

格式-3：IFFALSE/IFF 〈命令序列〉

作用：如果测试条件为假，则执行命令序列。否则，顺序执行后继命令。

IFT和IFF条件语句根据什么东西来判断条件是否成立呢？这就要用到测试命令TEST。

格式：TEST 〈条件〉

作用：将条件是否成立的测试结果提供给最接近的IFT或IFF命令使用。

例14.30 假设我们要根据n的值来确定是调用TR_1吗还是调用TR_2，用TEST和IFT或IFF来写就是：

```
? TO E14.30  :N ↵
  TEST  :N> 0 ↵
  IFT  TR1 ↵
  IFF  TR2 ↵
END
```

执行情况是这样的：

当N＞0时，TEST测试的结果为真，因而执行过程TR_1。

当N≤0时，TEST测试的结果为假，因而执行过程TR_2。

由此可见：IFT或IFF都必须同TEST联用。程序中如果没有TEST，则不能使用IFT或IFF。

在使用条件语句时必须注意：

① IFT、IFF或IF中的条件、命令序列只能写在同一行上。如果一行写不完，只能由机器自动换行而不能人为的用↲换行。例如：

```
IF  :A>100  THEN FD 50 RT 90 FD 20 RT 90
      FD 50 RT 90 FD 20 RT 90 ↲
```

虽然看起来占了两行，但机器内部却当成一行来处理。

② IFT和IFF都只能根据TEST测试的结果来判断条件是否成立。

③ 一个过程中可以使用多个TEST命令，其结果也只能为该过程专用。

④ TEST相当于只能存放真或假值的一个变量。

停止程序执行的语句是STOP，其格式是：

格式：STOP

作用：停止程序的执行。

例14.31 在例14.28中，假设边长超过80就停止程序的执行。递归过程改为：

```
? TO E14.31  :L  :A ↲
  HT ↲
  IF  :L>80 THEN STOP ↲
  FD  :L  RT  :A ↲
  E14.31  :L+3   :A ↲
  END ↲
```

? E14.31 0 90 ↲   执行结果如图14.36所示。

这个程序没有什么困难之处，请读者自己分析其运行情况。

14.3 数、字和表处理功能

LOGO语言不但有很强的作图功能，而且也提供了数、字和表处理的能力。

14.3.1 数和算术运算

LOGO语言中的数分为整数和实数两类，如：

整数：1988，325，－35，＋184

实数：4.73，－13.98，－123，＋3.0，3.

LOGO提供了四种基本的算术运算：＋－＊／，运算规则同数学中一样。为改变运算的优先顺序和避免运算产生二义性，在书写算术表达式时应当注意适当的成对使用圆括号。LOGO中算术表达式的概念同BASIC一样。例如：

(325＋84)＊23/5，　4096/24＋(5－374)＊(－3).

对于算术运算，系统是这样处理的：

① 两个整数进行加、减、乘运算，其结果总是整数；

② 整数和实数进行混合运算，其结果总是实数；

③ 两个数如果能够整除，其结果为整数。如果不能够整除，则结果为实数；

④ 整数的表示范围是-2^{31}～2^{31}；

⑤ 实数运算的精度是7位有效数字，绝对值不超过10^{38}。

LOGO语言中也可以用指数，其一般形式是：

〈数〉{E/N}〈指数〉

其中：数又称为尾数，它可以是整数，也可以是实数。E表示10的正整数次方，N表示10的负整数次方。看下面例子就清楚了。

例：　10.325E3　表示 $10.325_{10}{}^{3}$，即 10325。

　　　10.325N3　表示 $10.325_{10}{}^{-3}$，即 0.010325。

注意：LOGO语言中指数的正负是用E和N来表示的，而且指数只能是无符号整数。指数形式的数也属于实数。

2． 数的输出

格式：PRINT/PR 〈表达式〉

作用：将表达式的值显示在屏幕上。

这里的表达式类似BASIC表达式。

例如：? PRINT (54.25+183)/2.72 ↵

87.2243

? PRINT (17.42E5−3728.4)/(528+4096) ↵

375.924

当然、表达式中也可以有变量和函数。请参见14.4.1节

14.3.2　字和字处理命令

1． 什么是字？

由字母或数字等符号组成的一串字符称为字。字的标志是双引号”。如："STUDENT，"X5，"135

但要注意的是：LOGO中的字不同于BASIC中的字符串。组成字的符号可以是字母、数字和部份标点符号。而空格、各种括号、算术运算符是不能组成字的。如："A+B，"A*2，"A(i)均不是字。

对于字，除了可用PRINT输出以外，还有专门的字处理命令。

2． 字的处理

格式一1：FIRST 〈字〉

作用：取出字的第一个字符。

例：? PRINT FIRST "STUDENT ↵

S

格式－2：LAST 〈字〉

作用：取出字的最后那个字符。

例：? PRINT LAST "STUDENT ↵

T

格式－3：BUTFIRST/BF 〈字〉

作用：取出字中除第一个字符以外的其它字符。

例：? PRINT BF "STUDENT ↵

TUDENT

格式－4：BUTLAST/BL 〈字〉

作用：取出字中除最后那个字符以外的其它字符。

例：? PRINT BL "STUDENT ↵

STUDEN

如果一个字中只含有一个字符，那么取出其第一个（最后一个）字符时只能是空。如：? PRINT BF "X ↵

这是打印出一个空串。不含任何字符的字称为空字。

格式－5：WORD 〈字$_1$〉〈字$_2$〉…〈字n〉

作用：将给定的字进行连接，从而生成一个新的字。

例：? PRINT WORD "GOOD "BYE ↵

GOODBYE

14.3.3 表和表处理命令

1. 什么是表？

把若干个字组合在一起，并用方括号［］括起来，就组成一个表。表中的字不能带引号，每个字彼此间要用空格隔开。如：［This is a book］，［GOOD BYE］，

[AX_1 B_{32} XYZ123] 均是正确的表。这里的方括号可以看成是表的标志，它不是表中内容。表中每一个单独的字称为表的元素。

2. 表处理

对于表处理来说，字处理中的前四条命令同样适用于表处理，只是将字符换成表元素而已。

例：? PRINT FIRST [GOOD BYE] ↲

GOOD

? PRINT LAST [GOOD BYE] ↲

BYE

? PRINT BF [THIS IS A BOOK] ↲

IS A BOOK

? PRINT BL [THIS IS A BOOK] ↲

THIS IS A

在字处理中有一个WORD。相应地，表处理中有一个SENTENCE命令。

格式：SENTENCE/SE 〈字/表/数〉

作用：将若干字、表或数进行连接，从而构成一个新表。

例：? PRINT (SE [HELLO] "MY [FRIE-
ND]) ↲

HELLO MY FRIEND

对于表处理，还有两个命令：

格式：FPUT 〈字/表〉〈表〉

作用：将第一个〈字/表〉作为一个表元素放在第二个表中元素的最前面。

例：? FPUT "HELLO [MY FRIEND] ↲

RESULT：[HELLO MY FRIEND]

? FPUT [HELLO] [MY FRIEND] ↵

RESULT：[[HELLO] MY FRIEND]

格式：LPUT 〈字/表〉〈表〉

作用：将第一个〈字/表〉作为一个表元素放在第二个表中所有元素之后。

例：? LPUT "YOU [HOW ARE] ↵

RESULT：[HOW ARE YOU]

? LPUT [YOU] [HOW ARE] ↵

RESULT：[HOW ARE [YOU]]

14.4 数值计算功能

14.4.1 **表达式**

1. 数

数的概念和书写前面已经接触到了，我们可以简单地把它理解为日常用到的整数和实数就行了。

2. 变量

LOGO中也有变量。前面我们定义递归过程和有参过程时已经用到了。其概念同BASIC变量类似。变量名可以是一个字母或字母后再跟数字。在取变量名时必须注意：LOGO语言中已有固定含义的符号不能再作用户的变量名。

3. 函数

为了使用的方便，LOGO语言提供了一系列的函数。

函数名称和格式	功能
SIN :X	求X的正弦值。X为角度。

COS :X　　　求X的余弦值。X为角度。

ATAN :X :Y　求反正切函数值。X，Y分别表示对边和邻边的长度。

SQRT :X　　　求$\sqrt{X}$的平方根。X≥0。

ROUND :X　　将X四舍五入取整。

INTEGER :X　只取X的整数部份。

RANDOM :X　产生0～X之间的一个整数。X>0且为整数。

RANDOMIZE　产生随机数。

QUOTIENT :X :Y将X/Y的商取整。

REMAINDER :X :Y 求X/Y的余数。

4. 表达式

LOGO语言中的表达式可分成算术表达式和条件表达式。

① 算术表达式

算术表达式是用算术运算符（＋－＊／）或园括号将数、变量、数值函数按一定规则连接而成的式子。如：

:B＊ :B－4＊ :A＊ :C，（537＋SQRT 100)/ :A，(SIN :X＋ :Y)/(COS :X＋15)均是正确的算术表达式。从这里可以看出，在书写算术表达式时，表达式中的变量名之前一定要有冒号“：”。且冒号与之前的字符间至少有一个空格。这是和其它语言不同的地方。

② 条件表达式

条件表达式除了前面谈到的关系表达式以外，还可以用到逻辑非运算NOT，其作用是对其后的逻辑结果进行否定。例如：NOT 5＞3，NOT :L＝ :B，NOT（:B＊ :B－4＊ :A＊ :C＞0）。如果原来是真，加上否定过后就成

为假。

另外，LOGO还提供了类型判别条件，格式如下：

格式：NUMBER? 〈参数〉

作用：检查参数是否是数，如果是数，其值为真，否则为假。

格式：WORD? 〈参数〉

作用：检查参数是否为字，如果是字，其值为真，否则为假。

格式：LIST? 〈参数〉

作用：检查参数是否为表，如果是表，则其值为真，否则为假。

我们可以用这三种条件来判别一个参数的数据类型。

例14.32 判断X的数据类型（数、字、表）。

```
? TO E14.32 :X ↲
  IF NUMBER? :X THEN PRINT [NUMB-
  ER]
  IF WORD? :X THEN PRINT [WORD]
  IF LIST? :X THEN PRINT [LIST]
  END
? E14.32 139 ↲
  NUMBER            139既是数又是字。
  WORD
? E14.32 "ABC↲
  WORD
? E14.32 [GOOD BYE] ↲
  LIST
```

为使条件能够具备与或功能，LOGO还提供了另外两个条件表达式。

格式：ALLOF （表达式$_1$）（表达式$_2$）…（表达式$_n$）

作用：当所有表达式的值均为真时，该条件表达式为真，否则为假。

格式：ANYOF （表达式$_1$）（表达式$_2$）…（表达式$_n$）

作用：当某个表达式的值为真时，该条件表达式为真。当所有表达式的值均为假时，该条件表达式为假。

例：IF ALLOF (:A＞0) (:B＞ :C) (:C＜5) THEN…只有当三个关系表达式同时为真时，整个条件表达式的值才能为真。ALLOF (:A＞0) (:B＞ :C) (:C＜5)相当于BASIC语言中的逻辑表达式：A＞0 AND B＞C AND C＜5。

例：IF ANYOF (:A＞0) (:B＞ :C)(:C＜5) THEN…当三个关系表达式中只要有一个的值为真时，整个条件表达式的值就为真。ANYOF (:A＞0) (:B＞ :C) (:C＜5)相当于BASIC语言中的逻辑表达式：A＞0 OR B＞C OR C＜5。

14.4.2 **赋值命令和传送命令**

1．赋值命令

格式：MAKE "〈变量名〉〈表达式〉

作用：将表达式的值赋给指定变量。

其中：表达式可以是算术表达式、数、字或者表。

例：
```
? MAKE "X 5 +SQRT 100 ↵
? PRINT :X ↵
15
? MAKE "A [HOW ARE YOU] ↵
? PRINT :A ↵
```

HOW ARE YOU

在LOGO语言中，给变量赋值之前不必事先说明变量的类型，并允许同一变量在不同时刻赋与不同类型的数据。

例：
```
? MAKE "Y 135.72 ↲
? PRINT :Y ↲
135.72
? MAKE "Y "HOW ↲
? PRINT :Y ↲
HOW
? MAKE 'Y 3 > 5 ↲
? PRINT :Y ↲
FALSE （逻辑值假）
```

2. 传送命令

如果我们希望把一个过程的结果直接用于调用另一个过程，就必须用到传送命令（输出命令）。

格式：OUTPUT 〈表达式〉

作用：允许表达式的值输出到另一过程中。我们先来看下面的例子。

例：
```
? TO SS1 :X ↲
OUTPUT :X* :X ↲
END
? PRINT SS1 10 ↲
100
```

例：
```
? TO ADD1 :X :Y ↲
OUTPUT :X+ :Y ↲
END
? MAKE "D SS1 10 ↲
```

```
? PRINT  :D
100
? MAKE "D ADD1 (SS1  10)  100 ↵
? PRINT  :D ↵
200
```

从上述例子中我们看到了，用OUTPUT命令就可以把执行过程的结果送到一个变量中或直接作为另一过程的实在参数参与运算。请读者试一试，如果过程中没有OUTPUT命令，情况又如何呢？

OUTPUT命令也可用于字表处理中。

例：

```
? TO TR1  W ↵
  IF  :W = "  THEN STOP ↵
  PRINT  :W ↵
  TR1 BL  :W ↵
END
? TR1 "STUDENT ↵        (以下是运行结果)
STUDENT
STUDEN
STUDE
STUD
STU
ST
S
?
```

例：

```
? TO TR2  :X ↵
  OUTPUT WORD  :X  :X ↵
  END ↵
```

```
? TR1 TR2 "ME ↵       （以下是运行结果）
MEME
MEM
ME
M
?
```

在上述调用中，将TR_2 "ME的结果作为TR_1过程的输入量。如果TR_2过程中没有OUTPUT命令，那么TR_1 TR_2 "ME是不能直接使用TR_2 "ME作为输入量的。

注意：OUTPUT只能用于将执行过程后得到的值传送出来，我们可以认为其值就好象是存放在命名为OUTPUT的变量名中一样。如果需要把表达式的值传送到一个变量中，只能用MAKE命令。

14.4.3 如何编写数值计算程序

利用LOGO语言提供的条件语句、赋值语句、打印语句等就可以编写一些用于数值计算的简单程序了。

例14.33 求1＋2＋3＋…＋100的值.

```
? TO E14.331 :I :S ↵
  IF :I>100 PRINT :S STOP ↵
  MAKE "S :S+ :I ↵
  MAKE "I :I+1 ↵
  E14.331 :I :S ↵
  END ↵
  ? E14.331 1 0 ↵
  5050
```

该过程改成如下形式更为简单：

```
? TO E14.332  :I  :S ↵
  IF  :I>100 PRINT  :S  STOP ↵
  E14.332  :I+1  :S+ :I  ↵
  END
? E14.332  1  0 ↵
5050
```

例14.34 求$x^2+bx+c=0$的根。

根据一元二次方程的求根公式：

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

若b^2-4ac $\begin{cases} >0 & \text{有两个不等实根。} \\ =0 & \text{有两个相等实根。} \\ <0 & \text{共轭虚根。} \end{cases}$

如果有虚根，程序中X_1为实部，X_2为虚部。

```
? TO ROOL  :A  :B  :C ↵
  MAKE "D (:B *  :B- 4 *  :A *  :C) ↵
  TEST  :D> 0 ↵
  IFT MAKE "X1 ( -:B+SQRT :D)/(2 *  :A)
       MAKE "X2 ( -:B-SQRT :D)/(2 *  :A)
     MAKE  "G  " 实根  ↵
  TEST  :D< 0 ↵
  IFT MAKE  "X1 ( -:B ) / ( 2 *  :A )
     MAKE  "X2 ( SQRT  -:D ) / ( 2 *  :A )
     MAKE  "G "虚根 ↵
  TEST  :D= 0 ↵
  IFT MAKE  "X1 ( -:B ) / ( 2 * :A )
     MAKE  "X2  : X1
```

```
    MAKE "G "相等实根 ↵
(PRINT [X1=]  :X1 [X2=]  :X2  :G) ↵
END ↵
CTRL-C
```

在输入上述程序时，没有给出回车标志"↵"的行不能敲回车键。如果一个语句很长以致一行容纳不下，我们可以继续敲入，由系统自动换行（系统自动换行后屏幕上显示一个"!"符号）。

下面我们仍然用第八章中给定的数据来运行这段LOGO程序，其结果是一致的。

```
?ROOL  3   4   5 ↵
X1=-.666666  X2=1.10554     虚根
?ROOL  8   20  10 ↵
X1=-.690983  X2=-1.80902    实根
?ROOL  17  19  6 ↵
X1=-.558823  X2=.201637     虚根
?ROOL  24  56  43 ↵
X=-1.16667  X2=.656167      虚根
?ROOL  13  21  8 ↵
X1=-.615384  X2=-1.     实根
?ROOL  3   6   3 ↵
X1=-1.  X2=-1.     相等实根
?
```

14.5 LOGO基本命令一览表

为便于查找浏览，将已经介绍的LOGO命令列表如下：

命令格式	缩写	功能
·ASPECT X		设置屏幕纵横比例值。
ALLOF X_1 X_2…		对所有条件表达式求与。
ANYOF X_1 X_2…		对所有条件表达式求或。
BACK X	BK X	海龟回退X步。
BUTFIRST X	BF X	取字中除第一个字符以外的其它字符。
BUTLAST X	BL X	取字符中除最后字符以外的其它字符。
CLEARSCREEN	CS	清屏幕。
DRAW		清屏幕，海龟处于初始位置。
EDIT X	ED X	允许编辑过程X。
END		过程定义结束。
FIRST X		取第一个字符或第一个表元素。
FORWARDS X	FD X	海龟前进X步。
FPUT X_1 X_2		将X_1作为表元素放在X_2表中的第一个元素位置。
HEADING		测定转动角度。
HIDETURTLE	HT	将海龟隐藏起来。
HOME		将海龟置于起始位置。
IF X_1 〔THEN〕 X_2		X_1为真，执行X_2命令。
IFFALSE X	IFF X	测试条件为假，执行X命令。
IFTURE X	IFT X	测试条件为真，执行X命令。
LAST X		取出最后一个字符或元素。
LEFT X	LT X	海龟左转X度。

LIST? X		检测X是否为表，若是其值为真。
LPUT X_1 X_2		将X_1作为表元素放在X_2表中的最末尾。
MAKE "X_1 X_2		将X_2的值送到X_1中。
NUMBER? X		测定X是否为数，若是其值为真。
OUTPUT X		允许输出X的值。
PENDOWN	PD	海龟落下画笔。
PENUP	PU	海龟抬起画笔。
PRINT X	PR	屏幕显示X的值。
REPFAT X_1 〔X_2〕		将命令X_2重复执行X_1次。
RIGHT X	RT X	将海龟右转X度。
SETHEADING X	SETH X	将海龟转动X度。
SENTENCE X	SE X	将若干数、字、表连接成句子。
SETX X		将海龟水平移动X步。
SETXY X Y		将海龟移到（X，Y）座标位置。
SETY X		将海龟垂直移动X步。
SHOWTURTLE	ST	显现海龟。
STOP		停止执行。
TEST X		测试X条件的值。
TO X		定义或编辑过程X。
TOWARDS X Y		求Y轴与座标（X,Y）间的夹角。
WORD X_1 X_2…		将若干字进行连接。

WORD? X	检测X是否为字，若是其值为真。
XCOR	求海龟的X座标值。
YCOR	求海龟的Y座标值。

这里仅列出了一些主要的命令，不是中华学习机上LOGO的全部命令。

14.6 海龟作图程序选编

本节选编了一些典型图形的程序，读者可以上机试一试，检测在中华学习机上能否画出这些图形。除了利用示例中给出的参数调用过程外，自己还可以给出不同的参数来调用过程，无疑是有趣的。

14.6.1 利用正方形来构成的图形

例14.35

```
 ? TO RSQUA :L
REPEAT 4 [FD :L RT 90]
END
 ? TO MOVE :S
FD :S RT 90
FD :S LT 90
END
 ? TO STAI :S
RSQUA :S
MOVE :S
STAI :S
```

```
END
 ? STAI  30↲
```

结果如图14.5所示。

例14.36

```
 ? TO BOXES
  RSQUA 30
  MOVE 30
  RSQUA 20
  MOVE 20
  RSQUA 10
  MOUE 10
  RSQUA  5
  RT  180
  PU
  MOVE 60
  RT  180
  PD
 END
```

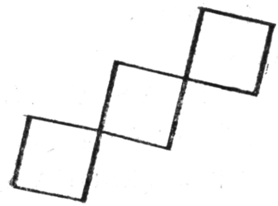


图14.50 STAI 30

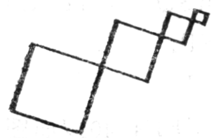


图14.51 BOXES

?BOXES↲ 如图14.51。

例14.37

```
 ? TO MBOXES
  BOXES
  FD 30 RT 90
  MBOXES
  END
```

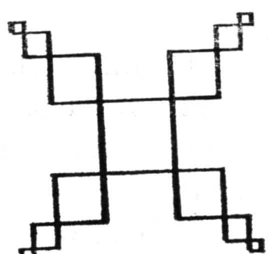


图14.52 MBOXES

```
? MBOXES ↲
```

执行结果如图14.52所示。

例14.38

```
 ? TO MSQU  :L
REPEAT 4 [REPE-
AT 4 [RSQUA :L RT
90] RT 45]
   HOME
   MSQU :L+ :L
   END
 ? MSQU  20↲
```

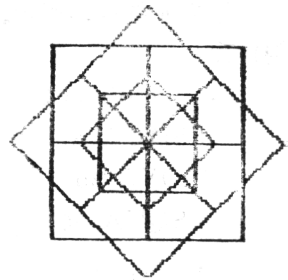


图14.53 MSQU 20

运行结果如图14.53所示。

例14.39

```
 ? TO SPIN :L
 RSQUA :L RT 20
 SPIN :L
END
 ? SPIN 40↲
```

如图14.54。

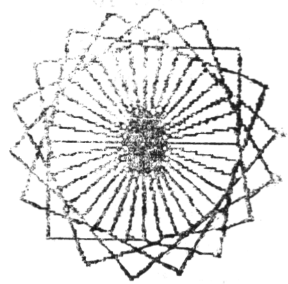


图14.54 SPIN 40

14.6.2 利用矩形来构成的图形

例14.40

```
 ? TO RECT :L :W
  FD :L RT 90 FD :W RT 90
  FD :L RT 90 FD :W RT 90
END
 ? TO FLOW
 RECT 80 20
```

```
RT  20
RECT 10   40
RT  20
FLOW
END
```

? FLOW ↵ 结果如图14.55所示。

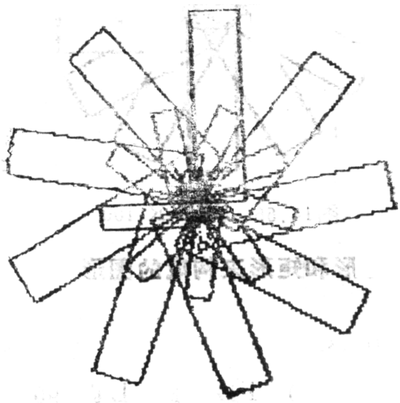


图14.55 FLOW

例14.41

```
? TO SPINR :S
IF :S<10 [STOP]
RECT :S 20
LT 30
SPINR  :S-5
END
? SPINR 100 ↵
```

执行结果如图14.56所示。

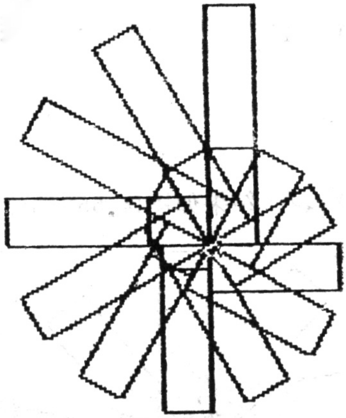


图14.56 SPINR 100

14.6.3 利用正方形和矩形来构成的图形

例14.42

```
? TO HOP :S
 FD :S RT 90 FD 3 LT 90
 END
? TO TELE :S
 IF :S<6 [STOP]
 RSQUA :S
 HOP :S
 TELE :S-6
 END
? TELE 30 ↵
```

结果如图14.57所示。

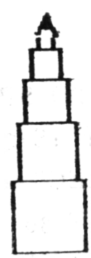


图14.57 TELE 30

例14.43

```
? TO ROCK
```

```
LT 30 FD 30 LT 120 FD 30
  END
? TO ROCKET
  RECT 80 30
  LT 90
  RECT 15 15
  BK 30
  RT 90
  RECT 15 15
  FD 80
  ROCK
  HT
  END
? ROCKET ↵
```



图14.58 ROCKET

结果如图14.58所示。

14.6.4 利用三角形来构成的图形

例14.44

```
? TO TRI :S
  REPEAT 3 [FD :S RT 120]
  END
? TO FLU :S
  IF :S<10 [STOP]
  TRI :S
  FLU :S-10
  END
? FLU 100 ↵
```

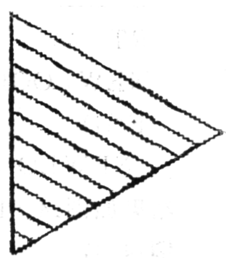


图14.59 FLU 100

结果如图14.59所示。

例14.45

```
? TO NEW :S
  LT 30  TRI :S RT 30
  END
? TO CREEP :S
  PU FD :S PD
  END
? TO LOOP :S
  NEW :S
  CREEP :S
  RT 60
  LOOP :S
  END
? LOOP 50 ↲
```

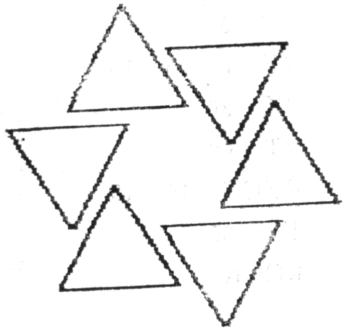


图14.60 LOOP 50

结果如图14.60所示。

例14.46

```
? TO NEWLOOP :S
  IF :S<20 [STOP]
  NEW :S
  CREEP :S/2
  RT 60
  NEWLOOP :S-5
  END
? NEWLOOP 80↲
```

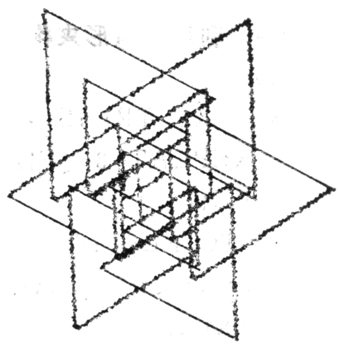


图14.61 NEWLOOP 80

结果如图14.61所示。

例14.47

```
? TO LANT
  LT 15 FD 30 LT 120 FD 15 BK 15
```

```
  RT  120  BK  30  RT  15
END
?  TO RANT
  RT 15 FD 30 RT 120 FD 15 BK 15
  LT  120  BK  30  LT  15
END
?  TO ANTS
  RANT  LANT
END
?  TO WING
  TRI 80  TRI  60 TRI 40 TRI 20
END
?  TO BUTT
RT  60
WING
RT  180
WING
RT  120
ANTS
RT  150
TRI  30
END
```

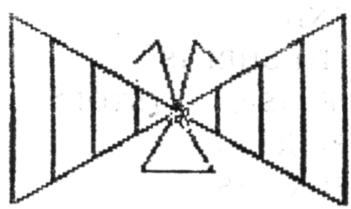


图14.62 BUTT

? BUTT 结果如图14.62所示。

14.6.5 利用弧和圆来构成的图形

例14.48

```
?  TO RCP  :R
  RT  5  FD :R*3.14159/18  RT  5
```

```
END
? TO LCP :R
  LT 5 FD :R*3.14159/18 LT 5
END
? TO RARC :R
  REPEAT 9 [RCP :R]
  END
? TO LARC :R
  REPEAT 9 [LCP :R]
  END
? TO RCIR :R
  REPEAT 36 [RCP :R]
  END
? TO LCIR :R
  REPEAT 36 [LCP :R]
  END
? TO SHRI :S
  IF :S<4 [STOP]
  RCIR :S
  SHRI :S-5
END
```

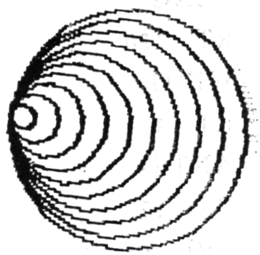


图14.63 SHRI 50

? SHRI 50 ↵ 结果如图14.63所示。

例14.49

```
? TO RSLI :S
  RCIR :S
  PU RT 90
  FD 10 LT 90
```

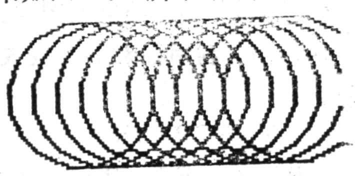


图14.64 RSLI 30

```
  PD
  RSLI :S
END
? RSLI 30↙
```
结果如图14.64所示。

例14.50

```
? TO SPINS :S
  RCIR :S
  RT 20
  SPINS :S
END
? SPINS 30↙
```

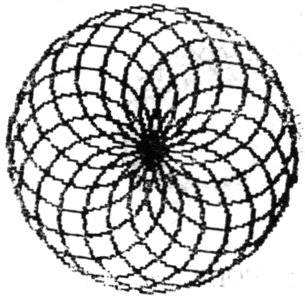


图14.65 SPINS 30

结果如图14.65所示。

例14.51

```
? TO RAY :S
  RARC :S LARC :S
  RARC :S LARC :S
END
? TO SUN :S
  RAY :S
  RT 160
  SUN :S
END
? SUN 25 ↙
```

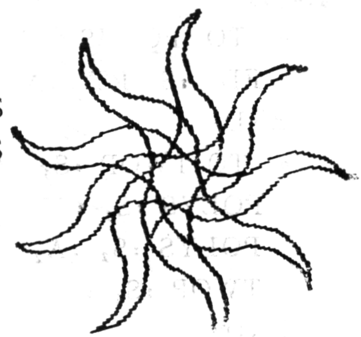


图14.66 SUN 25

结果如图14.66所示。

14.6.6 变化距离和角度来构成不同的图形

例14.52

```
? TO POLY :L :A
  FD :L RT :A
```

```
POLY :L:A
END
?  POLY 60 250
?  POLY 100 150
```

? POLY 60 250 结果如图14.67所示。

? POLY 100 150 结果如图14.68所示。

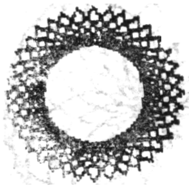


图14.67 POLY 60 250

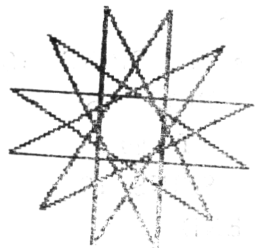


图14.68 POLY 100 150

例14.53

```
?  TO POLYS :S :A
  FD :S RT :A
END
?  TO TWOP :S1 :A1 :S2 :A2
  POLYS :S1 :A1
  POLYS :S2 :A2
  TWOP :S1 :A1 :S2 :A2
END
?  TWOP 30 90 50 160↵         如图14.69。
?  TWOP 80 80 80  80↵         如图14.70。
?  TWOP 30 90 50  135↵        如图14.71。
?  TWOP 30 60 90  100↵        如图14.72。
```

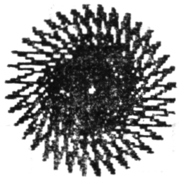


图14.69 TWOP 30 90 50 160

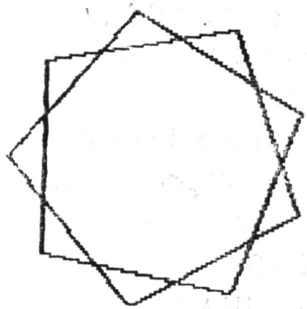


图14.70 TWOP 80 80 80 80

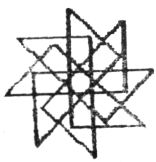


图14.71 TWOP 30 90 50 135

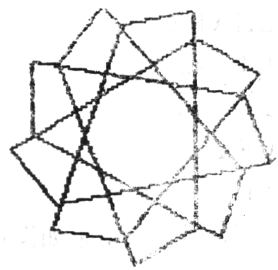


图14.72 TWOP 30 60 90 100

14.6.7 用一个图形来构成其它图形

例14.54

```
? TO SHAPE
  REPEAT 2 [FD 40 RT 90]
  REPEAT 2 [FD 20 RT 90]
  FD 40 RT 90
  REPEAT 2 [FD 10 RT 90]
  FD 20
END
```



图14.73 SHAPE

```
?  SHAPE ↵
```

结果如图14.73所示。

例14.55

```
?  TO SHAPE4
   REPEAT 4 [SHAPE]
   HT
END
?  SHAPE4 ↵
```

结果如图14.74所示。

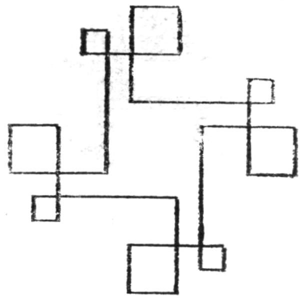


图14.74 SHAPE4

例14.56

```
?  TO CRYS
   SHAPE
   LT 45
   FD 70
   CRYS
END
?  CRYS
```

结果如图14.75所示。

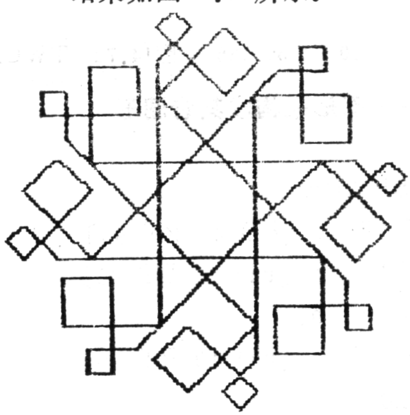


图14.75 CRYS

例14.57

```
? TO JENG
  SHAPE
  SHAPE
  LT 90
  JENG
END
? JENG ↵
```

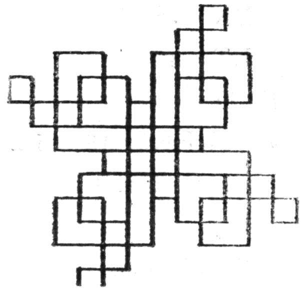


图14.76 JENG

结果如图14.76所示。

例14.58

```
? TO TREE1 :S :C
  IF :C=0 [STOP]
  LT 30
  FD :S*2
  TREE1 :S :C-1
  BK :S*2
  RT 60
  FD :S
  TREE1 :S :C-1
```

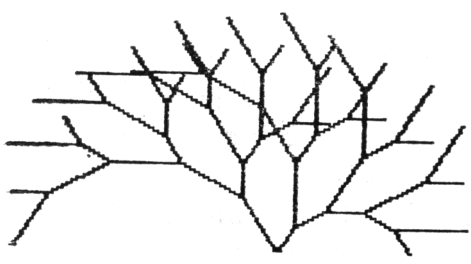


图14.77 TREE 1 15 5

```
  BK  :S
  LT  30
END
?  TREE1  15   5↵
```
结果如图14.77所示。

```
?  TO  TREE
  PU
  SETY -40
  PD
  TREE1 15  5
  BK  40
END
?   TREE↵
```
结果如图14.78所示。

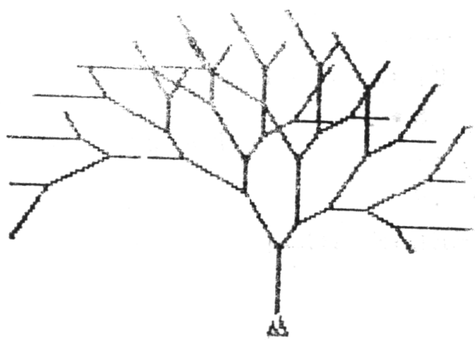


图14.78 TREE

例14.59

```
?  TO  VEE
  LT  45  FD  10  BK  10  RT  90
  FD  10  BK  10  LT  45
END
```

```
?  TO  BRAN
FD 15 VEE FD 15 VEE FD 10 BK 40
END
?  TO  BUSH
  LT  60
  REPEAT 6 [BRAN RT 20]
  BRAN
  LT  60
END
?  BUSH ↵
```



图14.79 BUSH

结果如图14.79。

例14.60

```
?  TO GRTREE
  FD  50
  BUSH
  BK  50
END
```

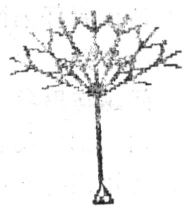


图14.80 GRTREE

```
?  GRTREE ↵
```

结果如图14.80

例14.61

```
?  TO  MOV
  PU
  RT  90
  FD  80
  LT  90
  PD
  END
?  TO TREES
```

```
  REPEAT 6[GRTREE  MOV]
END
? TREES ↲
```

结果如图14.81所示。



图14.81 TREES

请读者仔细推敲这些程序，并注意观察图形在屏幕上的绘制次序。

问题与思考

1. 如果我们希望在海龟初始位置以外的地方开始作图，可以用什么办法？

2. 海龟在屏幕上移动一步，其X座标值肯定增加1，这是真的吗？举例说明。

3. SETX，SETY，SETXY同FD和BK有何区别？SETHEADING和RT、LT又有何区别？

4. 用LOGO进行数值计算时如何解决问题所需要的循环（至少是一重循环）？

习　　题

编写绘制下列图形的程序。

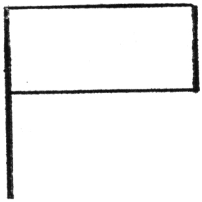


图 1

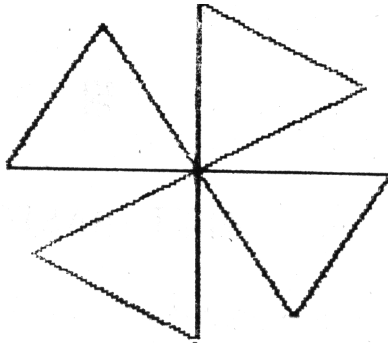


图 2

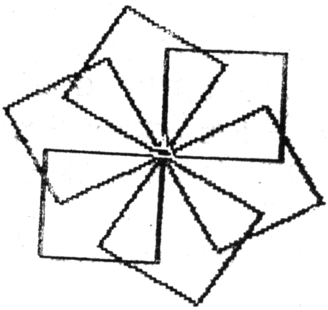


图 3

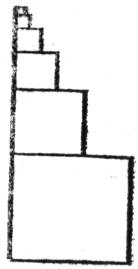


图 4

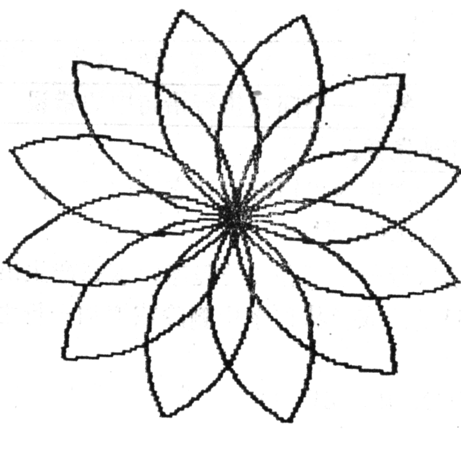


图 5

附　　录

附录I　BASIC出错信息表

在一个错误产生后，BASIC 返 回到命令一级以上，这由提示字符及一个闪烁的光标指出。变量和程序正文仍然是完整的，但是程序不能继续执行，并 且 所有的 GOSUB和FOR循环计数器都置成0，要在一个 运 行程序中避免这种中断，则可以把ONERR GOTO语句与错误 处 理例行程序结合起来使用。

当一个错误产生在一个立即执行语句中时，行号不被打印出来。出错信息的格式如下：

立即执行方式的语句　　？XX ERROR

间接执行方式的语句　　？XX ERROR INYY

在上面两个例子中，“XX”表示错误名，“YY”是产生错误的语句行号。一个间接执行语句中的错误，只有该语句被执行时才被发觉。

以下是可能产生的出错码以及它们的含义：

CAN'T CONTINUE

试图继续执行一个不存在的程序，或在产生出错后，试图继续执行程序，或者从程序中删除或增加一行之后，试图继续执行程序。

DIVISION BY ZERO

除数为零出错。

ILLEGAL DIRECT

你不能将INPUT，DEF FN，GET，DATA语句作为立即执行命令使用。

ILLEGAL QUANTITY

传送给数学函数或串函数的参数超出范围，产生ILLEGAL QUANTITY错误的情况可能是：

（a）负数为数组下标变量(例如：LETA（-1）=0)。

（b）用负数或零作为LOG的参数。

（c）用负数作SQR的参数。

（d）在A∧B中，A为负数，且B不是一个整数。

（e）在MID$，LEFT$，RIGHT$，WAIT，PEEK，POKE，TAB，SPC，ON…GOTO或作图的任一函数中使用了不恰当的参数。

NEXT WITHOUT FOR

在一个NEXT语句中的变量名与当前的FOR语句中的变量名不一致，或者无变量名的NEXT语句不与任何FOR语句相对应。

OUT OF DATA

执行一个READ语句时，程序中的所有DATA语句早已被读完，程序试图要读更多的数据，但程序中所包含的数据不够。

OUT OF MEMORY

以下的任何一种情况都会产生这种信息：程序太长；变量太多；FOR循环嵌套多于10层的深度；GOSUB的嵌套深度超过24层；表达式太复杂；圆括号嵌套深度多于36层；企图置的LOMEM：值太高；企图置的LOMEM：值低于当前值；企图置的HIMEM：值太低。

FORMULA TOO COMPLEX

执行到多于两条IF “XX” THEN形式的语句。

OVERFLOW

一个计算的结果太长，以致于不能用BASIC的数据格式，如果出现下溢出，给出一个零作为结果，并且继续执行，没有任何错误信息打印出来。

REDIM'D ARRAY

对一个数组两次使用DIM语句时，产生此错误。这个错误经常发生在这种情况下：如果直接使用象A（I）＝3这样的语句，则BASIC自动定义其标量长度为10，但在程序中又跟有如 DIM A（100）的语句，如果你想要找出一个数组定义语句所在的程序行，这个出错信息是很有用的。这只要在第一行上插入该数组的定义语句DIM，运行这个程序，则BASIC将告诉你原来的 DIM语句是什么位置上。

RETURN WITHOUT GOSUB

遇到了一条RETURN 语句，但是没有执行相应的GO-SUB语句。

STRING TOO LONG

试图通过使用连接运算产生一个多于255个字符的字符串。

BAD SUBSCRIPT

试图访问一个超出数组大小范围的数组元素，如果在一个数组的访问中使用了错误维数，这种出错信息亦产生。例如：当A已经用DIM A（2，2）定义了，但出现了LET A（1，1，1）＝Z的语句。

SYNTAX ERROR

在一个表达式中，丢掉了园括号，在一行中有不合法的

字符，不正确的标点符号等等，均会产生此信息。

TYPE MISMATCH

一个赋值语句的左边是一个数值变量，而右边是一个串，或者左边是一个串变量，而右边是一个数值型数据，另一种情况是一个函数的参数类型是一个串，但给出的却是一个数值型数据，或者反过来。

UNDEF'D STATEMENT

试图使用GOTO，GOSUB使THEN转到一个行号不存在的语句上去。

UNDEF'D FUNCTION

调用了还未定义的用户定义函数。

附录Ⅱ　DOS 3.3出错信息表

由于在命令或程序中使用DOS命令不当，会出现如下错误信息，这些错误信息的代码可由 BASIC语句 ERR=PEEK（222）得到。

错误代码	错误信息：	出错原因
1	LANGUAGE NOT AVAILABLE	没有装入整数BASIC解释程序，而使用了INIT命令。
2,3	RANGE ERROR	命令参数取值超出规定的范围。
4	WRITE PROTECTED	执行写操作时，软盘上贴有写保护。
5	END OF DATA	读操作中，读数据指

		针越过文本文件的末端。
6	FILE NOT FOUND	文件名拼错，没有找到该文件。
7	VOLUME MISMATCH	DOS命令使用中，卷号不匹配。
8	I/O ERROR	盘片被损坏，或软盘片未经过初始化，或未关闭驱动器的门。
9	DISK FULL	软盘上的文件个数超过额定数，或盘上数据已存满。
10	FILE LOCKED	试图对一个加有封锁的文件进行修改，删除，或重命名操作。
11	SYNTAX ERROR	DOS命令，参数标识符或逗号分隔符用错。或没有启动DOS操作系统，而试图使用DOS命令。
12	NOBUFFERS AVAILABLE	同时打开的文件个数超出 MAXFILES 命令中所规定的文件个数。
13	FILE TYPE	软盘文件的类型与DOS命令所操作的文件类型不匹配。

14	PROGRAM TOO LARGE	没有足够的内存空间来装入新的程序。
15	NOT DIRECT COMMAND	该命令必须在程序中作为语句来使用。

附录Ⅲ 区位码和汉字、符号对照表

1 区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 (SP) 、 。 · ˉ ˇ ¨ 〃 々 — ～ ‖ … ‘ ’ “ ” 〔 〕

20～39 〈 〉 《 》 「 」 『 』 〖 〗 【 】 ± × ÷ ∶ ∧ ∨ ∑ ∏

40～59 ∪ ∩ ∈ ∷ √ ⊥ ∥ ∠ ⌒ ⊙ ∫ ∮ ≡ ≌ ≈ ∽ ∝ ≠ ≮ ≯

60～79 ≤ ≥ ∞ ∵ ∴ ♂ ♀ ° ′ ″ ℃ ＄ ¤ ￠ ￡ ‰ § № ☆ ★

80～94 ○ ● ◎ ◇ ◆ □ ■ △ ▲ ※ → ← ↑ ↓ 〓

2 区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 1. 2. 3.

20～39 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. (1) (2) (3)

40～59 (4) (5) (6) (7) (8) (9) (10) (11) (12) (13) (14) (15) (16) (17) (18) (19) (20) ① ② ③

60～79 ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ (一) (二) (三) (四) (五) (六) (七) (八) (九) (十)

80～94 Ⅰ Ⅱ Ⅲ Ⅳ Ⅴ Ⅵ Ⅶ Ⅷ Ⅸ Ⅹ Ⅺ Ⅻ

区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

！ ” ＃ ￥ ％ ＆ ’ （ ） ＊ ＋ ， － ． ／ ０ １ ２ ３

４ ５ ６ ７ ８ ９ ： ； < ＝ > ？ ＠ Ａ Ｂ Ｃ Ｄ Ｅ Ｆ Ｇ

Ｈ Ｉ Ｊ Ｋ Ｌ Ｍ Ｎ Ｏ Ｐ Ｑ Ｒ Ｓ Ｔ Ｕ Ｖ Ｗ Ｘ Ｙ Ｚ ［

＼ ］ ＾ ＿ ｀ ａ ｂ ｃ ｄ ｅ ｆ ｇ ｈ ｉ ｊ ｋ ｌ ｍ ｎ ｏ

ｐ ｑ ｒ ｓ ｔ ｕ ｖ ｗ ｘ ｙ ｚ ｛ ｜ ｝ ￣

区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

ぁあぃいぅうぇえぉおかがきぎくぐけげこ

ごさざしじすずせぜそぞただちぢっつづてで

とどなにぬねのはばぱひびぴふぶぷへべぺほ

ぼぽまみむめもゃやゅゆょよらりるれろゎわ

ゐゑをん

5区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01～19		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク	グ	ケ	ゲ	コ
20～39	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ
40～59	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ
60～79	ボ	ポ	マ	ミ	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
80～94	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ													

6区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01～19		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ
20～39	Υ	Φ	Χ	Ψ	Ω									α	β	γ	δ	ε	ζ	η
40～59	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	φ	χ	ψ	ω			
60～79																				
80～94																				

7 区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19　А Б В Г Д Е Ё Ж З И Й К Л М Н О П Р С

20～39　Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я

40～59　а б в г д е ё ж з и й

60～79　к л м н о п р с т у ф х ц ч ш щ ь ы ъ э

80～94　ю я

8 区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19　ā á ǎ à ē é ě è ī í ǐ ì ō ó ǒ ò ū ú ǔ

20～39　ù ǖ ǘ ǚ ǜ ü e ㄅ ㄆ ㄇ

40～59　ㄈ ㄉ ㄊ ㄋ ㄌ ㄍ ㄎ ㄏ ㄐ ㄑ ㄒ ㄓ ㄔ ㄕ ㄖ ㄗ ㄘ ㄙ ㄚ ㄛ

60～79　ㄜ ㄝ ㄞ ㄟ ㄠ ㄡ ㄢ ㄣ ㄤ ㄥ ㄦ ㄧ ㄨ ㄩ

80～94

9区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19					─	━	│	┃	┄	┅	┆	┇	┈	┉	┊	┋	┌	┍	┎	┏
02~39	┐	┑	┒	┓	└	┕	┖	┗	┘	┙	┚	┛	├	┝	┞	┟	┠	┡	┢	┣
40~59	┤	┥	┦	┧	┨	┩	┪	┫	┬	┭	┮	┯	┰	┱	┲	┳	┴	┵	┶	┷
60~90	┸	┹	┺	┻	┼	┽	┾	┿	╀	╁	╂	╃	╄	╅	╆	╇	╈	╉	╊	╋
87~94																				

16区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		a 啊	阿	ai 埃	挨	哎	唉	哀	皑	癌	蔼	矮	艾	碍	爱	隘	an 鞍	氨	安	俺	A
20~39	按	暗	岸	胺	案	ang 肮	昂	盎	ao 凹	敖	熬	翱	袄	傲	奥	懊	澳	ba 芭	捌	扒	B
40~59	叭	吧	笆	八	疤	巴	拔	跋	靶	把	耙	坝	霸	罢	爸	bai 白	柏	百	摆	佰	
60~79	败	拜	稗	abn 斑	班	搬	扳	般	颁	板	版	扮	拌	伴	瓣	半	办	绊	bang 邦	帮	
80~94	梆	榜	膀	绑	棒	磅	蚌	镑	傍	谤	bao 苞	胞	包	褒	剥						

17区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		薄	雹	保	堡	饱	宝	抱	报	暴	豹	鲍	爆	bei 杯	碑	悲	卑	北	辈	背
20~39	贝	钡	倍	狈	备	惫	焙	被	ben 奔	苯	本	笨	beng 崩	绷	甭	泵	蹦	迸	bi 逼	鼻
40~59	比	鄙	笔	彼	碧	蓖	蔽	毕	毙	毖	币	庇	痹	闭	敝	弊	必	辟	壁	臂
60~79	避	陛	bian 鞭	边	编	贬	扁	便	变	卞	辨	辩	辫	遍	biao 标	彪	膘	表	bie 鳖	憋
80~94	别	瘪	bin 彬	斌	濒	滨	宾	摈	bing 兵	冰	柄	丙	秉	饼	炳					

18区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		病	并	bo 玻	菠	播	拨	钵	波	博	勃	搏	铂	箔	伯	帛	舶	脖	膊	渤
20~39	泊	驳	bu 捕	卜	哺	补	埠	不	布	步	簿	部	怖	ca 擦	cai 猜	裁	材	才	财	睬
40~59	踩	采	彩	菜	蔡	can 餐	参	蚕	残	惭	惨	灿	cang 苍	舱	仓	沧	藏	cao 操	糙	槽
60~79	曹	草	ce 厕	策	侧	册	测	cen 层	蹭	cha 插	叉	茬	茶	查	碴	搽	察	岔	差	诧
80~94	chai 拆	柴	豺	chan 搀	掺	蝉	馋	谗	缠	铲	产	阐	颤	chang 昌	猖					

C

19区

		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
													chao							
01~19		场	尝	常	长	偿	肠	厂	敞	畅	唱	倡	超	抄	钞	朝	嘲	潮	巢	吵
		che						chen										cheng		
20~39	炒	车	扯	撤	掣	彻	澈	郴	臣	辰	尘	晨	忱	沉	陈	趁	衬	撑	称	城
													chi							
40~50	橙	成	呈	乘	程	惩	澄	诚	承	逞	骋	秤	吃	痴	持	匙	池	迟	弛	驰
									chong					chou						
60~79	耻	齿	侈	尺	赤	翅	斥	炽	充	冲	虫	崇	宠	抽	酬	畴	踌	稠	愁	筹
						chu														
80~94	仇	绸	瞅	丑	臭	初	出	橱	厨	躇	锄	雏	滁	除	楚					

20区

		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
								chuai	chuan							chuang				
01~19		础	储	矗	搐	触	处	揣	川	穿	椽	传	船	喘	串	疮	窗	幢	床	闯
		chui					chun							chuo		ci				
20~39	创	吹	炊	捶	锤	垂	春	椿	醇	唇	淳	纯	蠢	戳	绰	疵	茨	磁	雌	辞
								cong						cou	cu				cuan	
40~59	慈	瓷	词	此	刺	赐	次	聪	葱	囱	匆	从	丛	凑	粗	醋	簇	促	蹿	篡
		cui								cun			cuo						da	
60~79	窜	摧	崔	催	脆	瘁	粹	淬	翠	村	存	寸	磋	撮	搓	措	挫	错	搭	达 D
						dai														
80~94	答	瘩	打	大	呆	歹	傣	戴	带	殆	代	贷	袋	待	逮					

21区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
		dan															dang		
01～19	怠	耽	担	丹	单	郸	掸	胆	旦	氮	但	惮	淡	诞	弹	蛋	当	挡	党
			dao												de			deng	
20～39	荡	档	刀	捣	蹈	倒	岛	祷	导	到	稻	悼	道	盗	德	得	的	蹬	灯	登
					di														
40～59	等	瞪	凳	邓	堤	低	滴	迪	敌	笛	狄	涤	翟	嫡	抵	底	地	蒂	第	帝
				dian															diao
60～79	弟	递	缔	颠	掂	滇	碘	点	典	靛	垫	电	佃	甸	店	惦	奠	淀	殿	碉
									die										
80～94	叼	雕	凋	刁	掉	吊	钓	调	跌	爹	碟	蝶	迭	谍	叠				

22区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
	ding									diu	dong								
01～19	丁	盯	叮	钉	顶	鼎	锭	定	订	丢	东	冬	董	懂	动	栋	侗	恫	冻
		dou							du										
20～39	洞	兜	抖	斗	陡	豆	逗	痘	都	督	毒	犊	独	读	堵	睹	赌	杜	镀	肚
				duan						dui				dun					
40～59	度	渡	妒	端	短	锻	段	断	缎	堆	兑	队	对	墩	吨	蹲	敦	顿	囤	钝
			duo												e				
60～79	盾	遁	掇	哆	多	夺	垛	躲	朵	跺	舵	剁	惰	堕	蛾	峨	鹅	俄	额	讹 E
								en	er										
80～94	娥	恶	厄	扼	遏	鄂	饿	恩	而	儿	耳	尔	饵	[illegible]	二				

23区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
			fa								fan									
01～19		贰	发	罚	筏	伐	乏	阀	法	珐	藩	帆	番	翻	樊	矾	钒	繁	凡	烦
									fang										fei	
20～39	反	返	范	贩	犯	饭	泛	坊	芳	方	肪	房	防	妨	仿	访	纺	放	菲	非
											fen									
40～59	啡	飞	肥	匪	诽	吠	肺	废	沸	费	芬	酚	吩	氛	分	纷	坟	焚	汾	粉
						feng														
60～79	奋	份	忿	愤	粪	丰	封	枫	蜂	峰	锋	风	疯	烽	逢	冯	缝	讽	奉	凤
	fo	fou	fu																	
80～94	佛	否	夫	敷	肤	孵	扶	拂	辐	幅	氟	符	伏	俘	服					

F

24区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01～19		浮	涪	福	袱	弗	甫	抚	辅	俯	釜	斧	脯	腑	府	腐	赴	副	覆	赋
														ga		gai				
20～39	复	傅	付	阜	父	腹	负	富	讣	附	妇	缚	咐	噶	嘎	该	改	概	钙	盖
		gan											gang							
40～59	溉	干	甘	杆	柑	竿	肝	赶	感	秆	敢	赣	冈	刚	钢	缸	肛	纲	岗	港
		gao										ge								
60～79	杠	篙	皋	高	膏	羔	糕	搞	镐	稿	告	哥	歌	搁	戈	鸽	胳	疙	割	革
									gei	gen		geng								
80～94	葛	格	蛤	阁	隔	铬	个	各	给	根	跟	耕	更	庚	羹					

G

25区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19

gong (04) gou (19)

埂耿梗工攻功恭龚供躬公宫弓巩汞拱贡共钩

20～39

gu (28)

勾沟苟狗垢构购够辜菇咕箍估沽孤姑鼓古蛊骨

40～59

guo (46) guai (52) guan (55)

谷股故顾固雇刮瓜剐寡挂褂乖拐怪棺关官冠观

60～79

guang (66) gui (69)

管馆罐惯灌贯光广逛瑰规圭硅归龟闺轨鬼诡癸

80～94

gun (85) guo (88) ha (94)

桂柜跪贵刽辊滚棍锅郭国果裹过哈

26区 01 02 03 04 05 06 07 08 09 10 11 02 13 14 15 16 17 18 19

01～19

hai (01) han (08)

骸孩海氦亥害骇酣憨邯韩含涵寒函喊罕翰撼

20～36

hang (27) hao (30)

捍旱憾悍焊汗汉夯杭航壕嚎豪毫郝好耗号浩呵

40～59

hei (57) hen (59)

喝荷菏核禾和何合盒貉阂河涸赫褐鹤贺嘿黑痕

60～79

heng (63) hong (68) hou (76)

很狠恨哼亨横衡恒轰哄烘虹鸿洪宏弘红喉侯猴

80～94

hu (84)

吼厚候后呼乎忽瑚壶葫胡蝴狐糊湖

27区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		弧	虎	唬	护	互	沪	户	hua 花	哗	华	猾	滑	画	划	化	话	huai 槐	徊	怀
20~39	淮	坏	huan 欢	环	桓	还	缓	换	患	唤	痪	豢	焕	涣	宦	幻	huang 荒	慌	黄	磺
40~59	蝗	簧	皇	凰	惶	煌	晃	幌	恍	谎	hui 灰	挥	辉	徽	恢	蛔	回	毁	悔	慧
60~79	卉	惠	晦	贿	秽	会	烩	汇	讳	诲	绘	hun 荤	昏	婚	魂	浑	混	huo 豁	活	伙
80~94	火	获	或	惑	霍	货	祸	ji 击	圾	基	机	畸	稽	积	箕					

J

28区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		肌	饥	迹	激	讥	鸡	姬	绩	缉	吉	极	棘	辑	籍	集	及	急	疾	汲
20~39	即	嫉	级	挤	几	脊	己	蓟	技	冀	季	伎	祭	剂	悸	济	寄	寂	计	记
40~59	既	忌	际	妓	继	纪	jia 嘉	枷	夹	佳	家	加	荚	颊	贾	甲	钾	假	稼	价
60~79	架	驾	嫁	jian 歼	监	坚	尖	笺	间	煎	兼	肩	艰	奸	缄	茧	检	柬	碱	硷
80~94	拣	捡	简	俭	剪	减	荐	槛	鉴	践	贱	见	键	箭	件					

29区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01～19		健	舰	剑	饯	渐	溅	涧	建	jiang 僵	姜	将	浆	江	疆	蒋	桨	奖	讲	匠
20～39	酱	降	jiao 蕉	椒	礁	焦	胶	交	郊	浇	骄	娇	嚼	搅	铰	矫	侥	脚	狡	角
40～59	饺	缴	绞	剿	教	酵	轿	较	叫	窖	jie 揭	接	皆	秸	街	阶	截	劫	节	桔
60～79	杰	捷	睫	竭	洁	结	解	姐	戒	藉	芥	界	借	介	疥	诫	届	jin 巾	筋	斤
80～94	金	今	津	襟	紧	锦	仅	谨	进	靳	晋	禁	近	烬	浸					

30区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01～19		尽	劲	jing 荆	兢	茎	睛	晶	鲸	京	惊	精	粳	经	井	警	景	颈	静	境
20～39	敬	镜	径	痉	靖	竟	竞	净	jiong 炯	窘	jiu 揪	究	纠	玖	韭	久	灸	九	酒	厩
40～59	救	旧	臼	舅	咎	就	疚	ju 鞠	拘	狙	疽	居	驹	菊	局	咀	矩	举	沮	聚
60～79	拒	据	巨	具	距	踞	锯	俱	句	惧	炬	剧	juan 捐	鹃	娟	倦	眷	卷	绢	jue 撅
80～94	攫	抉	掘	倔	爵	觉	决	诀	绝	jun 均	菌	钧	军	君	峻					

31区

K

		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		俊	竣	浚	郡	骏	ka 喀	咖	卡	咯	kai 开	揩	楷	凯	慨	kan 刊	堪	勘	坎	砍
20~39	看	kang 康	慷	糠	扛	抗	亢	炕	kao 考	拷	烤	靠	ke 坷	苛	柯	棵	磕	颗	科	壳
40~59	咳	可	渴	克	刻	客	课	ken 肯	啃	垦	恳	keng 坑	吭	kong 空	恐	孔	控	kou 抠	口	扣
60~79	寇	ku 枯	哭	窟	苦	酷	库	裤	kua 夸	垮	挎	跨	胯	kuai 块	筷	侩	快	kuan 宽	款	kuang 匡
80~94	筐	狂	框	矿	眶	旷	况	kui 亏	盔	岿	窥	葵	奎	魁	傀					

32区

L

		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		馈	愧	溃	kun 坤	昆	捆	困	kuo 括	扩	廓	阔	la 垃	拉	喇	蜡	腊	辣	啦	lai 莱
20~39	来	赖	lan 蓝	婪	栏	拦	篮	阑	兰	澜	谰	揽	览	懒	缆	烂	滥	lang 琅	榔	狼
40~59	廊	郎	朗	浪	lao 捞	劳	牢	老	佬	姥	酪	烙	涝	le 勒	乐	lei 雷	镭	蕾	磊	累
60~79	儡	垒	擂	肋	类	泪	leng 棱	楞	冷	li 厘	梨	犁	黎	篱	狸	离	漓	理	李	里
80~94	鲤	礼	莉	荔	吏	栗	丽	厉	励	砾	历	利	傈	例	俐					

33区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
									lia	lian									
01~19	痢	立	粒	沥	隶	力	璃	哩	俩	联	莲	连	镰	廉	怜	涟	帘	敛	脸
					liang										liao				
20~39	链	恋	炼	练	粮	凉	梁	粱	良	两	辆	量	晾	亮	谅	撩	聊	僚	疗	燎
									lie				lin						
40~59	寥	辽	潦	了	撂	镣	廖	料	列	裂	烈	劣	猎	琳	林	磷	霖	临	邻	鳞
					ling														liu
60~79	淋	凛	赁	吝	拎	玲	菱	零	龄	铃	伶	羚	凌	灵	陵	岭	领	另	令	溜
											long								
80~94	琉	榴	硫	馏	留	刘	瘤	流	柳	六	龙	聋	咙	笼	窿				

34区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
					lou						lu								
01~19	隆	垄	拢	陇	楼	娄	搂	篓	漏	陋	芦	卢	颅	庐	炉	掳	卤	虏	鲁
												lu							
20~39	麓	碌	露	路	赂	鹿	潞	禄	录	陆	戮	驴	吕	铝	侣	旅	履	屡	缕	虑
						luan						lue		lun					
40~59	氯	律	率	滤	绿	峦	挛	孪	滦	卵	乱	掠	略	抡	轮	伦	仑	沦	纶	论
	luo												ma						
60~79	萝	螺	罗	逻	锣	箩	骡	裸	落	洛	骆	络	妈	麻	玛	码	蚂	马	骂	嘛 M
		mai						man											
80~94	吗	埋	买	麦	卖	迈	脉	瞒	馒	蛮	满	蔓	曼	慢	漫				

35区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
		mang						mao											
01～19	谩	芒	茫	盲	氓	忙	莽	猫	茅	锚	毛	矛	铆	卯	茂	冒	帽	貌	贸
	me	mei																men	
20～39	么	玫	枚	梅	酶	霉	煤	没	眉	媒	镁	每	美	昧	寐	妹	媚	门	闷	们
	meng								mi										
40～59	萌	蒙	檬	盟	锰	猛	梦	孟	眯	醚	靡	糜	迷	谜	弥	米	秘	觅	泌	蜜
			mian									miao								mie
60～79	密	幂	棉	眠	绵	冕	免	勉	娩	缅	面	苗	描	瞄	藐	秒	渺	庙	妙	蔑
		min						ming						miu	mo				
80～94	灭	民	抿	皿	敏	悯	闽	明	螟	鸣	铭	名	命	谬	摸				

36区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
																	mou		
01～19	摹	蘑	模	膜	磨	摩	魔	抹	末	莫	墨	默	沫	漠	寞	陌	谋	牟	某
	mu															na				
20～39	拇	牡	亩	姆	母	墓	暮	幕	募	慕	木	目	睦	牧	穆	拿	哪	呐	钠	那 N
			nai					nan			nang	nao					ne	nei		nen
40～59	娜	纳	氖	乃	奶	耐	奈	南	男	难	囊	挠	脑	恼	闹	淖	呢	馁	内	嫩
	neng	ni											nian							niang
60～79	能	妮	霓	倪	泥	尼	拟	你	匿	腻	逆	溺	蔫	拈	年	碾	撵	捻	念	娘
		niao		nie							nin	ning							
80～94	酿	鸟	尿	捏	聂	孽	啮	镊	镍	涅	您	柠	狞	凝	宁				

37区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19
niu nong nu nu nuan nue nuo
拧泞牛扭钮纽脓浓农弄奴努怒女暖虐疟挪懦

20～39
o ou pa pai
糯诺哦欧鸥殴藕呕偶沤啪趴爬帕怕琶拍排牌徘 OP

40～59
pan pong pao
湃派攀潘盘磐盼畔判叛乓庞旁耪胖抛咆刨炮袍

60～79
pei pen peng
跑泡呸胚培裴赔陪配佩沛喷盆砰抨烹澎彭蓬棚

80～49
pi
硼篷膨朋鹏捧碰坯砒霹批披劈琵毗

38区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19
pian piao pie
啤脾疲皮匹痞僻屁譬篇偏片骗飘漂瓢票撇瞥

20～39
pin ping po
拼频贫品聘乒坪苹萍平凭瓶评屏坡泼颇婆破魄

40～59
pou pu qi
迫粕剖扑铺仆莆葡菩蒲埔朴圃普浦谱曝瀑期欺 Q

50～79
栖戚妻七凄漆柒沏其棋奇歧畦崎脐齐旗祈祁骑

80～94
qia
起岂乞企启契砌器气迄弃汽泣讫掐

39区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

qian
01～19 恰洽牵扦钎铅千迁签仟谦乾黔钱钳前潜遣浅

qiang qiao
20～39 谴堑嵌欠歉枪呛腔羌墙蔷强抢橇锹敲悄桥瞧乔

qie qin
40～59 侨巧鞘撬翘峭俏窍切茄且怯窃钦侵亲秦琴勤芹

qing qiong qiu
60～79 擒禽寝沁青轻氢倾卿清擎晴氰情顷请庆琼穷秋

qu
80～94 丘邱球求囚酋泅趋区蛆曲躯屈驱渠

40区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

quan quə
01～19 取娶龋趣去圈颧权醛泉全痊拳犬券劝缺炔瘸

qun ran rang rao re
20～39 却鹊榷确雀裙群然燃冉染瓤壤攘嚷让饶扰绕惹 R

ren reng ri rong
40～59 热壬仁人忍韧任认刃妊纫扔仍日戎茸蓉荣融熔

rou ru ruan rui
60～79 溶容绒冗揉柔肉茹蠕儒孺如辱乳汝入褥软阮蕊

run ruo sa sai san
80～94 瑞锐闰润若弱撒洒萨腮鳃塞赛三叁 S

41区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19
sang sao se sen seng sha
伞散桑嗓丧搔骚扫嫂瑟色涩森僧莎砂杀沙刹

20～39
shai shan
纱傻啥煞筛晒珊苫杉山删煽衫闪狭擅赡膳善汕

40～59
shang shao
扇缮墒伤商赏晌上向裳梢捎稍烧芍勺韶少哨邵

60～79
she shen
绍奢赊蛇舌舍赦摄射慑涉社设砷申呻伸身深娠

80～94
sheng
绅神沈审婶甚肾慎渗声生甥牲升绳

42区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19
shi
省盛剩胜圣师失狮施湿诗尸虱十石拾时什食

20～39
蚀实识史矢使屎驶始式示士世柿事拭誓逝势是

40～59
shou
嗜噬适仕侍释饰氏市恃室视试收手首守寿授售

60～79
shu
受瘦兽蔬枢梳殊抒输叔舒淑疏书赎孰熟薯暑曙

80～94
署蜀黍鼠属术述树束戍竖墅庶数漱

43区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19
shua shuai shuan shuang shui shun
恕刷耍摔衰甩帅栓拴霜双爽谁水睡税吮瞬顺

20～39
shuo si
舜说硕朔烁斯撕嘶思私司丝死肆寺嗣四伺似饲

40～59
song sou su
巳松耸怂颂送宋讼诵搜艘擞嗽苏酥俗素速粟僳

60～79
suan sui sun
塑溯宿诉肃酸蒜算虽隋随绥髓碎岁穗遂隧祟孙

80～94
suo ta
损笋蓑梭唆缩琐索锁所塌他它她塔

T

44区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19
tai tan
獭挞蹋踏胎苔抬台泰酞太态汰坍摊贪瘫滩坛

20～39
tang
檀痰潭谭谈坦毯袒碳探叹炭汤塘搪堂棠膛唐糖

40～59
tao te teng
倘躺淌趟烫掏涛滔绦萄桃逃淘陶讨套特藤腾疼

60～79
ti tian
誊梯剔踢锑提题蹄啼体替嚏惕涕剃屉天添填田

80～94
tiao tie ting
甜恬舔腆挑条迢眺跳贴铁帖厅听烃

45区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 汀廷停亭庭挺艇(tong)通桐酮瞳同铜彤童桶捅筒统

20～39 痛(tuo)偷投头透(tu)凸秃突图徒途涂屠土吐兔(tuan)湍团(tui)推颓

40～59 腿蜕褪退(tun)吞屯臀(tuo)拖托脱鸵陀驮驼椭妥拓唾(wa)挖哇 W

60～79 蛙洼娃瓦袜(wai)歪外(wan)豌弯湾玩顽丸烷完碗挽晚皖惋

80～94 宛婉万腕(wang)汪王亡枉网往旺望忘妄(wei)威

46区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 巍微危韦违桅围唯惟为潍维苇萎委伟伪尾纬

20～39 未蔚味畏胃喂魏位渭谓尉慰卫(wen)瘟温蚊文闻纹吻

40～59 稳紊问(weng)嗡翁瓮(wo)挝蜗涡窝我斡卧握沃(wu)巫呜钨乌污

60～79 诬屋无芜梧吾吴毋武五捂午舞伍侮坞戊雾晤物

80～94 勿务悟误(xi)昔熙析西硒矽晰嘻吸锡牺 X

47区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

10~19 稀息希悉膝夕惜熄烯溪汐犀檄袭席习媳喜铣

20~39 洗系隙戏细(xia)瞎虾匣霞辖暇峡侠狭下厦夏吓(xian)掀锨

40~59 先仙鲜纤咸贤衔舷闲涎弦嫌显险现献县腺馅羡

60~79 宪陷限线(xiang)相厢镶香箱襄湘乡翔祥详想响享项巷

80~94 橡像向象(xiao)萧硝霄削哮嚣销消宵淆晓

48区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01~19 小孝校肖啸笑效(xie)楔些歇蝎鞋协挟携邪斜胁谐

20~39 写械卸蟹懈泄泻谢屑(xin)薪芯锌欣辛新忻心信衅(xing)星

40~59 腥猩惺兴刑型形邢行醒幸杏性姓(xiong)兄凶胸匈汹雄

60~79 熊(xiu)休修羞朽嗅锈秀袖绣(xu)墟戌需虚嘘须徐许蓄酗

80~94 叙旭序畜恤絮婿绪续(xuan)轩喧宣悬旋玄

49区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 选癣眩绚靴(xue)薛学穴雪血勋(xun)熏循旬询寻驯巡殉

20～39 汛训讯逊迅压(Ya)押鸦鸭呀丫芽牙蚜崖衙涯雅哑亚 Y

40～59 讶焉(Yan)咽阉烟淹盐严研蜒岩延言颜阎炎沿奄掩眼

60～79 衍演艳堰燕厌砚雁唁彦焰宴谚验殃(yang)央鸯秧杨扬

80～94 佯疡羊洋阳氧仰痒养样漾邀(yao)腰妖瑶

50区 01 02 03 04 05 06 07 18 09 10 11 12 13 14 15 16 17 18 19

01～19 摇尧遥窑谣姚咬舀药要耀椰(ye)噎耶爷野冶也页

20～39 掖业叶曳腋夜液一(yi)壹医揖铱依伊衣颐夷遗移仪

40～59 胰疑沂宜姨彝椅蚁倚已乙矣以艺抑易邑屹亿役

60～79 臆逸肄疫亦裔意毅忆义益溢诣议谊译异翼翌绎

80～94 茵(yin)荫因殷音阴姻吟银淫寅饮尹引隐

51区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

ying
01～19 印英樱婴鹰应缨莹萤营荧蝇迎赢盈影颖硬映

yo yong you
20～39 哟拥佣臃痈庸雍踊蛹咏泳涌永恿勇用幽优悠忧

yu
40～59 尤由邮铀犹油游酉有友右佑釉诱又幼迂淤于盂

60～79 榆虞愚舆余俞逾鱼愉渝渔隅予娱雨与屿禹宇语

80～91 羽玉域芋郁吁遇喻峪御愈欲狱育誉

52区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

yuan
01～19 浴寓裕预豫驭鸳渊冤元垣袁原援辕园员圆猿

yue yun
20～39 源缘远苑愿怨院曰约越跃钥岳粤月悦阅耘云郧

za zai zan
40～59 匀陨允运蕴酝晕韵孕匝砸杂栽哉灾宰载再在咱 Z

zang zao
60～79 攒暂赞赃脏葬遭糟凿藻枣早澡蚤躁噪造皂灶燥

ze zei zen zeng zha
80～94 责择则泽贼怎增憎曾赠扎喳渣札轧

53区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~91		铡	闸	眨	栅	榨	咋	乍	炸	诈	zhai 摘	斋	宅	窄	债	寨	zhan 瞻	毡	詹	粘
20~39	沾	盏	斩	辗	崭	展	蘸	栈	占	战	站	湛	绽	zhang 樟	章	彰	漳	张	掌	涨
40~59	杖	丈	帐	账	仗	胀	瘴	障	zhao 招	昭	找	沼	赵	照	罩	兆	肇	召	zhe 遮	折
60~79	哲	蛰	辙	者	锗	蔗	这	浙	zhen 珍	斟	真	甄	砧	臻	贞	针	侦	枕	疹	诊
80~94	震	振	镇	阵	zheng 蒸	挣	睁	征	狰	争	怔	整	拯	正	政					

54区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~11		帧	症	郑	证	zhi 芝	枝	支	吱	蜘	知	肢	脂	汁	之	织	职	直	植	殖
20~39	执	值	侄	址	指	止	趾	只	旨	纸	志	挚	掷	至	致	置	帜	峙	制	智
40~59	秩	稚	质	炙	痔	滞	治	窒	zhong 中	盅	忠	钟	衷	终	种	肿	重	仲	众	zhou 舟
60~79	周	州	洲	诌	粥	轴	肘	帚	咒	皱	宙	昼	骤	zhu 珠	株	蛛	朱	猪	诸	诛
80~94	逐	竹	烛	煮	拄	瞩	嘱	主	著	柱	助	蛀	贮	铸	筑					

55区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 住注祝驻 zhua 抓爪 zhuai 拽 zhuan 专砖转撰赚篆 zhuang 桩庄装妆撞壮

20～39 状 zhui 椎锥追赘坠缀 zhun 谆准 zhuo 捉拙卓桌琢茁酌啄着灼浊

40～59 zi 兹咨资姿滋淄孜紫仔籽滓子自渍字 zong 鬃棕踪宗综

60～79 总纵 zou 邹走奏揍 zu 租足卒族祖诅阻组 zuan 钻纂 zui 嘴醉最罪

80～94 zun 尊遵 zuo 昨左佐柞做作坐座

56区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 亍丌兀丐廿卅丕亘丞鬲孬噩丨禺八匕乇夭爻

20～39 卮氐囟胤馗毓睾鼗丶亟鼐乜乩亓芈孛啬嘏仄厍

40～59 厝厣厥厮靥赝匚叵匦匮匾赜卦卣刂刈刎刭刳刿

60～79 剀剌剞剡剜蒯剽劂劁劐劓冂罔亻仃仉仂仨仡仫

80～94 仞伛仳伢佤仵伥伧伉伫佞佧攸佚佝

57区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01～19		佟	佗	伲	伽	佶	佴	侑	侉	侃	侏	佾	佻	侪	佼	侬	侔	俦	俨	俪
20～39	俅	俚	俣	俜	俑	俟	俸	倩	偌	俳	倬	倏	倮	倭	俾	倜	倌	倥	倨	偾
40～59	偃	偕	偈	偎	偬	偻	傥	傧	傩	傺	僖	儆	僭	僬	僦	僮	儇	儋	全	氽	人入
60～79	佘	佥	俎	龠	汆	籴	兮	巽	黉	馘	冁	夔	勹	匍	訇	匐	凫	夙	兕	亠	八勺 儿乀
80～94	兖	亳	衮	袤	亵	脔	裒	禀	嬴	蠃	羸	冫	冱	冽	冼						

58区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01～19		凇	一	冢	冥	讠	讦	讧	讪	讴	讵	讷	诂	诃	诋	诏	诎	诒	诓	诔	冂讣
20～39	诖	诘	诙	诜	诟	诠	诤	诨	诩	诮	诰	诳	诶	诹	诼	诿	谀	谂	谄	谇
40～59	谌	谏	谑	谒	谔	谕	谖	谙	谛	谘	谝	谟	谠	谡	谥	谧	谪	谫	谮	谯
60～79	谲	谳	谵	谶	巾	卺	阝	阢	阡	阱	阪	阽	阼	陂	陉	陔	陟	陧	陬	陲	巾 阝
80～94	陴	隈	隍	隗	隰	邗	邛	邝	邙	邬	邡	邴	邳	邶	邺						

59区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01～19	邸	邰	郏	郅	邾	郐	郄	郇	郓	郦	郢	郜	郗	郛	郫	郯	郾	鄄	鄢	
20～39	鄞	鄣	鄱	鄯	鄹	酃	酆	刍	奂	劢	劬	劭	劾	哿	勐	勖	勰	叟	燮	矍	力刀 又
40～59	廴	冂	凼	鬯	厶	弁	畚	巯	坌	垩	垡	塾	墼	壅	壑	圩	圬	圪	圳	圹	廴匚 厶土
60～79	圮	圯	坜	圻	坂	坩	垅	坫	垆	坼	坻	坨	坭	坶	坳	垭	垤	垌	垲	埏	
80～94	垧	垴	垓	垠	埕	埘	埚	埙	埒	垸	埴	埴	埸	埤	埝						

60区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01～19	堋	堍	埽	埭	堀	堞	堙	塄	堠	塥	塬	墁	墉	墚	墀	馨	鼙	懿	艹	卄土
20～39	艽	艿	芏	芊	芨	芄	芎	芑	芗	芙	芫	芸	芾	芰	苈	苊	苣	芘	芷	芮	
40～59	苋	苌	苁	芩	芴	芡	芪	芟	苄	苎	芤	苡	茉	苷	苤	茏	茇	苜	苴	苒	
60～79	苘	茌	苻	苓	茑	茚	茆	茔	茕	苠	苕	茜	荑	荛	荜	茈	莒	茼	茴	茱	
80～94	莛	荞	茯	荏	荇	荃	荟	荀	茗	荠	茭	茺	茳	荦	荥						

61区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 荨茛荩荬荪荭荮莰荸莳莴莠莪莓莜莅荼莶莩

20～39 荽莸荻莘莞莨莺莼菁萁菥菘堇萘萋菝菽菖萜萸

40～59 萑萆菔菟萏萃菸菹菪菅菀萦菰菡葜葑葚葙葳蒇

60～79 蒈葺蒉葸萼葆葩葶蒌蒎萱葭蓁蓍蓐蓦蒽蓓蓊蒿

80～94 蒺蓠蒡蒹蒴蒗蓥蓣蔌甍蔸蓰蔹蔟蔺

62区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 蕖蔻蓿蓼蕙蕈蕨蕤蕞蕺瞢蕃蕲蕻薤薨薇薏蕹

20～39 薮薜薅薹薷薰藓藁藜藿蘧蘅蘩蘖蘼廾弈夼奁耷 廾大

40～59 奕奚奘匏尢尥尬尴扌扪抟抻拊拚拗拮挢拶挹捋 尢扌

60～79 捃掭揶捱捺掎掴捭掬掊捩掮掼揲揸揠揿揄揞揎

80～94 摒揆掾摅摁搋搛搠搌搦搡摞撄摭撖

63区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 摺撷撸撙撺擀擐擗擤擢攉攥攮弋忒甙弑卟叱 弋口

20～39 叽叩叨叻吒吖吆呋呒呓呔呖呃吡呗呙吣吲咂咔

40～59 呷呱呤咚咛咄呶呦咝哐咭哂咴哒咧咦哓哔呲咣

60～79 哕咻咿哌哙哚哜咩咪咤哝哏哞唛哧唠哽唔哳唢

80～94 唣唏唑唧唪啧喏喵啉啭啁啕唿啐唼

94区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 唷啖啵啶啷唳唰啜喋嗒喃喱喹喈喁喟啾嗖喑

20～39 啻嗟喽喾喔喙嗪嗷嗉嘟嗑嗫嗬嗔嗦嗝嗄嗯嗥嗲

40～59 嗳嗌嗍嗨嗵嗤辔嘞嘈嘌嘁嘤嘣嗾嘀嘧嘭噘嘹噗

60～79 嘬噍噢噙噜噌噔嚆噤噱噫噻噼嚅嚓嚯囔囗囝囡 口

80～94 囵囫囹囿圄圊圉圜帏帙帔帑帱帻帼 巾

65区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 帷幄幔幛幞幡岌屺岍岐岖岈岘岙岑岚岜岵岢 山

20～39 岽岬岫岱岣峁岷峄峒峤峋峥崂崃崧崦崮崤崞崆

40～59 崛嵘崾崴崽嵬嵛嵯嵝嵫嵋嵊嵩嵴嶂嶙嶝豳嶷巅

60～79 彳彷徂徇徉後徕徙徜徨徭徵徼衢 彡犭 犰犴犷犸 彳 彡犭

80～94 狃狁狎狍狒狨狯狩狲狴狷猁狳猃狺

66区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19 狻猗猓猡猊猞猝猕猢猹猥猬猸猱獐獍獗獠獬 夂 饣

20～39 獯獾舛夥飧夤夂饣饧饨饩饪饫饬饴饷饽馀馄馇

40～59 馊馍馐馑馓馔馕庀庑庋庖庥庠庹庵庾庳赓廒廑 广

60～79 廛廨廪膺忄忉忖忏怃忮怄忡忤忾怅怆忪忭忸怙 忄

80～94 怵怦怛怏怍怩怫怊怿怡恸恹恻恺恂

67区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19	恪 恽 悖 悚 悭 悝 悃 悒 悌 悛 惬 悻 悱 惝 惘 惆 惚 悴 愠
20～39	愦 愕 愣 惴 愀 愎 愫 慊 慵 憬 憔 憧 憷 懔 懵 忝 隳 闩 闫 闱
40～59	闳 闵 闶 闼 闾 阃 阄 阆 阈 阊 阋 阌 阍 阏 阒 阕 阖 阗 阙 阚
60～79	丬 爿 戕 氵 汔 汜 汊 沣 沅 沐 沔 沌 汨 汩 汴 汶 沆 沩 泐 泔
80～94	沭 泷 泸 泱 泗 沲 泠 泖 泺 泫 泮 沱 泓 泯 泾

68区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01～19	洹 洧 洌 浃 浈 洇 洄 洙 洎 洫 浍 洮 洵 洚 浏 浒 浔 洳 涑
20～39	浯 涞 涠 浞 涓 涔 浜 浠 浼 浣 渚 淇 淅 淞 渎 涿 淠 渑 淦 淝
40～59	淙 渖 涫 渌 涮 渫 湮 湎 湫 溲 湟 溆 湓 湔 渲 渥 湄 滟 溱 溘
60～79	滠 漭 滢 溥 溧 溽 溻 溷 滗 溴 滏 溏 滂 溟 潢 潆 潇 漤 漕 滹
80～94	漯 漶 潋 潴 漪 漉 漩 澉 澍 澌 潸 潲 潼 潺 濑

69区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		潍	澧	澹	澶	濂	濡	濮	濞	濠	濯	瀚	瀣	瀛	瀹	瀵	灏	灞	宀	宄	宀
20~39	宕	宓	宥	宸	甯	骞	搴	寤	寮	褰	寰	蹇	謇	辶	迓	迕	迥	迮	迤	迩	辶
40~59	迦	迳	迨	逅	逄	逋	逦	逑	逍	逖	逡	逦	逵	逶	逭	逯	遄	遑	遒	遐	遨
60~79	遘	遢	遛	暹	遴	遽	邂	邈	邃	邋	彐	彗	彖	彘	尻	咫	屐	屙	孱	屣	彐 尸
80~94	屦	羼	弪	弩	弭	艴	弼	鬻	中	妁	妃	妍	妩	妪	妣						弓 中 文

70区		01	02	03	04	05	09	07	08	09	01	11	12	13	14	15	19	17	19	19	
10~19		妗	姊	妫	妞	妤	姒	妲	妯	姗	妾	娅	娆	姝	娈	姣	姘	姹	娌	娉	
20~39	娲	娴	娑	娣	娓	婀	婧	婊	婕	娼	婢	婵	胬	媪	媛	婷	婺	媾	嫫	媲	
40~59	嫒	嫔	媸	嫠	嫣	嫱	嫖	嫦	嫘	嫜	嬉	嬗	嬖	嬲	嬷	孀	尕	尜	孚	孥	小子
90~79	孳	孑	孓	孢	驵	驷	驸	驺	驿	驽	骀	骁	骅	骈	骊	骐	骒	骓	骖	骘	马
80~94	骛	骜	骝	骟	骠	骢	骣	骥	骧	纟	纡	纣	纥	纨	纩						纟

71区 01 02 03 04 05 09 07 08 09 10 11 12 13 14 15 19 17 11 89

01～19	纭纰纾绀绁绂绉绋绌绐绔绗绛绠绡绨绫绮绯	
20～39	绱绲缍绶绺绻绾缁缂缃缇缈缋缌缏缑缒缗缙缜	
40～59	缛缟缡缢缣缤缥缦缧缪缫缬缭缯缰缱缲缳缵幺	幺
90～79	畿巛甾邕玎玑玮玢玟珏珂珑玷玳珀珉珈珥珙顼	巛 王
80～94	琊珩珧珞玺珲琏琪瑛琦琥琨琰琮琬	

72区 01 02 03 04 05 09 07 08 09 10 11 12 13 14 15 19 17 18 19

10～19	琛琚瑁瑜瑗瑕瑙瑷瑭瑾璜璎璀璁璇璋璞璨璩	
20～39	璐璧瓒璺韪韫韬杌杓杞杈杩枥枇杪杳枘枧杵枨	韦 木
40～59	枞枭枋杷杼柰栉柘栊柩枰栌柙枵柚枳柝栀柃枸	
90～79	柢栎柁柽栲栳桠桡桎桢桄桤梃栝桕桦桁桧桀栾	
80～94	棨桉栩梵梏桴桷梓桫棂楮棼椟椠棹	

73区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	19	17	18	19	
10~19		樛	桱	椋	桲	楗	棣	椐	棱	椹	楠	楂	楝	榄	楫	榀	榘	楸	椴	槌	
20~39	榇	榈	槎	榉	楦	楣	楹	榛	榧	榻	榫	榭	槔	榱	槁	槊	槟	榕	槠	榍	犬歹车
40~59	槿	樯	槭	樗	樘	橥	槲	橄	樾	檠	橐	橛	樵	檎	橹	樽	樨	橘	橼	檑	
90~79	檐	檩	檗	檫	猷	獒	殁	殂	殇	殄	殒	殓	殍	殚	殛	殡	殪	轫	轭	轱	
80~94	轲	轳	轵	轶	轸	轷	轹	轺	轼	轾	辁	辂	辄	辇	辋						

74区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		辍	辎	辏	辘	辚	軎	戋	戗	戛	戟	戢	戡	戥	戤	戬	臧	瓯	瓴	瓿	戈瓦
20~39	甏	甑	甓	攴	旮	旯	旰	昊	昙	杲	昃	昕	昀	炅	曷	昝	昴	昱	昶	昵	支日
40~59	耆	晟	晔	晁	晏	晖	晡	晗	晷	暄	暌	暧	暝	暾	曛	曜	曦	曩	贲	贳	贝
60~79	贶	贻	贽	赀	赅	赆	赈	赉	赇	赍	赕	赙	觇	觊	觋	觌	觎	觏	觐	觑	见
80~94	牮	犟	牝	牦	牯	牾	牿	犄	犋	犍	犏	犒	挈	挲	掰						牛手

75区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		毦	氂	毶	毪	毳	毽	毵	毹	氅	氇	氆	氍	氕	氘	氙	氚	氡	氩	氤	毛气
20~39	氪	氲	攵	敕	敫	牍	牒	牖	爰	虢	刖	肟	肜	肓	肼	朊	肽	肱	肫	肭	攵片月
40~59	肴	肷	胧	胨	胩	胪	胛	胂	胄	胙	胍	胗	朐	胝	胫	胱	胴	胭	脍	脎	
60~79	胲	胼	朕	脒	豚	脶	脞	脬	脘	脲	腈	腌	腓	腴	腙	腚	腱	腠	腩	腼	
80~94	腽	腭	腧	塍	媵	膈	膂	膑	滕	膣	膪	臌	朦	臊	膻						

76区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		臁	臜	欤	欷	欹	歃	歆	歙	飑	飒	飓	飕	飙	殳	彀	穀	毂	觳	斐	欠风殳文
20~39	齑	斓	於	旆	旄	旃	旌	旎	旒	旖	炀	炜	炖	炝	炻	烀	炷	炫	炱	烨	方火
40~59	烊	焐	焓	焖	焯	焱	煳	煜	煨	煅	煲	煊	煸	煺	熘	熳	熵	熨	熠	燠	灬户礻
60~79	燔	燧	燹	爝	爨	灬	焘	煦	熹	戾	戽	扃	扈	扉	礻	祀	祆	祉	祛	祜	心
80~94	祓	祚	祢	祗	祠	祯	祧	祺	禅	禊	禚	禧	禳	忑	忐						

77区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		怼	恝	恚	恧	恁	恙	恣	悫	愆	愍	慝	憩	憝	懋	懑	戆	肀	聿	沓	聿 水
20~39	泶	淼	矶	矸	砀	砉	砗	砘	砑	斫	砭	砜	砝	砹	砺	砻	砟	砼	砥	砬	石
40~59	砣	砩	硎	硭	硖	硗	砦	硐	硇	硌	硪	碛	碓	碚	碇	碜	碡	碣	碲	碹	
60~79	碥	磔	磙	磉	磬	磲	礅	磴	礓	礤	礞	礴	龛	黹	黻	黼	盱	眄	眍	盹	龙 业 目
80~94	眇	眈	眚	眢	眙	眭	眦	眵	眸	睐	睑	睇	睃	睚	睨						

78区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		睢	睥	睿	瞍	睽	瞀	瞌	瞑	瞟	瞠	瞰	瞵	瞽	町	畀	畎	畋	畈	畛	田
20~39	畲	畹	疃	罘	罡	罟	詈	罨	罴	罱	罹	羁	罾	盍	盥	蠲	钅	钆	钇	钋	罒 皿 钅
40~59	钊	钌	钍	钏	钐	钔	钗	钕	钚	钛	钜	钣	钤	钫	钪	钭	钬	钯	钰	钲	
60~79	钴	钶	钷	钸	钹	钺	钼	钽	钿	铄	铈	铉	铊	铋	铌	铍	铎	铐	铑	铒	
80~94	铕	铖	铗	铙	铘	铛	铞	铟	铠	铢	铤	铥	铧	铨	铪						

79区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01～19	铩	铫	铮	铯	铳	铴	铵	铷	铹	铼	铽	铿	锃	锂	锆	锇	锉	锊	锍
20～39	锎	锏	锒	锓	锔	锕	锖	锘	锛	锝	锞	锟	锢	锪	锫	锩	锬	锱	锲 锴
40～59	锶	锷	锸	锼	锾	锿	镂	锵	镄	镅	镆	镉	镌	镎	镏	镒	镓	镔	镖 镗
60～79	镘	镙	镛	镞	镟	镝	镡	镢	镤	镥	镦	镧	镨	镩	镪	镫	镬	镯	镱 镲
80～94	镳	锺	矧	矬	雉	秕	秭	秣	秫	稆	嵇	稃	稂	稞	稔				矢禾

80区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01～19	稹	稷	穑	黏	馥	穰	皈	皎	皓	皙	皤	瓞	瓠	甬	鸠	鸢	鸨	鸩	鸪 白瓜用鸟
20～39	鸫	鸬	鸲	鸱	鸶	鸸	鸷	鸹	鸺	鸾	鹁	鹂	鹄	鹆	鹇	鹈	鹉	鹋	鹌 鹎
40～59	鹑	鹕	鹗	鹚	鹛	鹜	鹞	鹣	鹦	鹧	鹨	鹩	鹪	鹫	鹬	鹱	鹭	鹳	疒 疔 疒
60～79	疖	疠	疝	疬	疣	疳	疴	疸	痄	疱	疰	痃	痂	痖	痍	痣	痨	痦	痤 痫
80～94	痧	瘃	痱	痼	痿	瘐	瘀	瘅	瘌	瘗	瘊	瘥	瘘	瘕	瘙				

81区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		瘗	瘼	瘢	瘠	癀	瘭	瘰	瘿	瘵	癃	瘾	瘳	癍	癞	癔	癜	癖	癫	癯	
20~39	翊	竦	穸	穹	窀	窆	窈	窕	窦	窠	窬	窨	窭	窳	衤	衩	衲	衽	衿	袂	立穴衤
40~59	袢	裆	袷	袼	裉	裢	裎	裣	裥	裱	褚	裼	裨	裾	裰	褡	褙	褓	褛	褊	
60~79	褴	褫	褶	襁	襦	襻	疋	胥	皲	皴	矜	耒	耔	耖	耜	耠	耢	耥	耦	耧	疋皮矛耒
80~49	耩	耨	耱	耋	耵	聃	聆	聍	聒	聩	聱	覃	顸	颀	颃						老耳西页

82区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	19	17	18	19	
01~19		颉	颌	颍	颏	颔	颚	颛	颞	颟	颡	颢	颥	颦	虍	虔	虬	虮	虿	虺	虍虫
20~39	虼	虻	蚨	蚍	蚋	蚬	蚝	蚧	蚣	蚪	蚓	蚩	蚶	蛄	蚵	蛎	蚰	蚺	蚱	蚯	
40~59	蛉	蛏	蚴	蛩	蛱	蛲	蛭	蛳	蛐	蜓	蛞	蛴	蛟	蛘	蛑	蜃	蜇	蛸	蜈	蜊	
90~79	蜍	蜉	蜣	蜻	蜞	蜥	蜮	蜚	蜾	蝈	蜴	蜱	蜩	蜷	蜿	螂	蜢	蝽	蝾	蝻	
80~94	蝠	蝰	蝌	蝮	螋	蝓	蝣	蝼	蝤	蝙	蝥	螓	螯	螨	蟒						

83区 01 02 03 04 05 09 07 08 06 10 11 12 13 14 15 19 17 18 19

01~19	螟螈螅螭螗螃螫蟥螬螵螳蟋蟓螽蟑蟀蟊蟛蟪	
20~39	蟠蟮蠖蠓蟾蠊蠛蠡蠹蠼缶罂罄罅舐竺竽笈笃笄	缶舌 竹
40~59	笕笊笫笏筇笸笪笙笮笱笠笥笤笳笾笞筘筚筅筵	
90~79	筌筝筠筮筻筢筲筱箐箦箧箸箬箝箨箅箪箜箢箫	
80~94	箴篑篁篌篝篚篥篦篪簌篾篼簏簖簋	

84区 01 02 03 04 05 09 07 08 09 10 11 12 13 14 15 16 17 18 19

01~19	簟簪簦簸籁籀臾舁舂舄臬衄舡舢舣舭舯舨舫	臼自 血舟
20~39	舸舻舳舴舾艄艉艋艏艚艟艨衾袅袈裘裟襞羝羟	衣羊
40~59	羧羯羰羲籼敉粑粝粜粞粢粲粼粽糁糇糌糍糈糅	米
90~79	糗糨艮暨羿翎翕翥翡翦翩翮翳糸絷綦綮繇纛麸	艮羽 系麦
80~45	麴赳趄趔趑趱赧赭豇豉酊酐酎酏酤	走赤 豆酉

85区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01~19	酢 酡 酰 酪 酯 酽 酾 酲 酴 酹 醌 醅 醐 醍 醑 醢 醣 醪 醭	
20~39	醮 醯 醵 醴 醺 豕 鹾 趸 跫 踅 蹙 蹩 趵 趿 趼 趺 跄 跖 跗 跚	豕 卤 足
41~59	跞 跎 跏 跛 跆 跬 跷 跸 跣 跹 跻 跤 踉 跽 踔 踝 踟 踬 踮 踣	
90~79	踯 踺 蹀 踹 踵 踽 踱 蹉 蹁 蹂 蹑 蹒 蹊 蹰 蹶 蹼 蹯 蹴 躅 躏	
80~94	躔 躐 躜 躞 豸 貂 貊 貅 貘 貔 斛 觖 觞 觚 觜	豸 角

86区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 19 17 18 19

01~19	觥 觫 觯 訾 謦 靓 雩 雳 雯 霆 霁 霈 霏 霎 霪 霭 霰 霾 龀	青 齿 言 雨
20~39	龃 龅 龆 龇 龈 龉 龊 龌 黾 鼋 鼍 隹 隼 隽 雎 雒 瞿 雠 銎 銮	隹 黾 金
40~99	鋈 錾 鍪 鏊 鎏 鐾 鑫 鱿 鲂 鲅 鲆 鲇 鲈 稣 鲋 鲎 鲐 鲑 鲒 鲔	鱼
90~79	鲕 鲚 鲛 鲞 鲟 鲠 鲡 鲢 鲣 鲥 鲦 鲧 鲨 鲩 鲫 鲭 鲮 鲰 鲱 鲲	
80~54	鲳 鲴 鲵 鲶 鲷 鲺 鲻 鲼 鲽 鳄 鳅 鳆 鳇 鳊 鳋	

87区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01～19		鳌	鳍	鳎	鳏	鳐	鳓	鳔	鳕	鳗	鳘	鳙	鳜	鳝	鳟	鳢	靼	鞅	鞑	鞒	革
20～39	鞔	鞯	鞫	鞣	鞲	鞴	骱	骰	骷	鹘	骶	骺	骼	髁	髀	髅	髂	髋	髌	髑	骨
40～59	魅	魃	魇	魉	魈	魍	魑	飨	餍	餮	饕	饔	髟	髡	髦	髯	髫	髻	髭	髹	鬼食 髟
60～79	鬈	鬏	鬓	鬟	鬣	麽	麾	縻	麂	麇	麈	麋	麒	鏖	麝	麟	黛	黜	黝	黠	麻鹿 黑
80～94	黟	黢	黩	黧	黥	黪	黯	鼢	鼬	鼯	鼹	鼷	鼽	鼾	齄						鼠鼻

为了使计算机知识早日在我国得到普及，我部应用户要求，特邀请四川大学计算机系韩仲清，廖兴祥老师在百忙中为我们编写了这本书。意在让广大的用户和计算机爱好者能够更早的了解和使用中华学习机。我部自经营中华学习机以来，得到了广大用户的大力支持和信赖。在此，谨借本书致谢。今后如果你在学习和使用中华学习机时遇到困难，三开元是你最好的朋友。我们备有中华学习机上所用的工具软件、教学软件、游戏软件、游戏磁带数百种及齐全的外围设备，随时供你选用，并免费提供清单。函索必寄。

成都市三开元电脑部

地址：成都市大科甲巷27号

如何使用中华学习机

成都三开元电脑部